Progetto di

Luigi Nardi

in collaborazione con

SELCON elettronica

Doppio visualizzatore di temperatura:

2µS1T2

2µS1T2

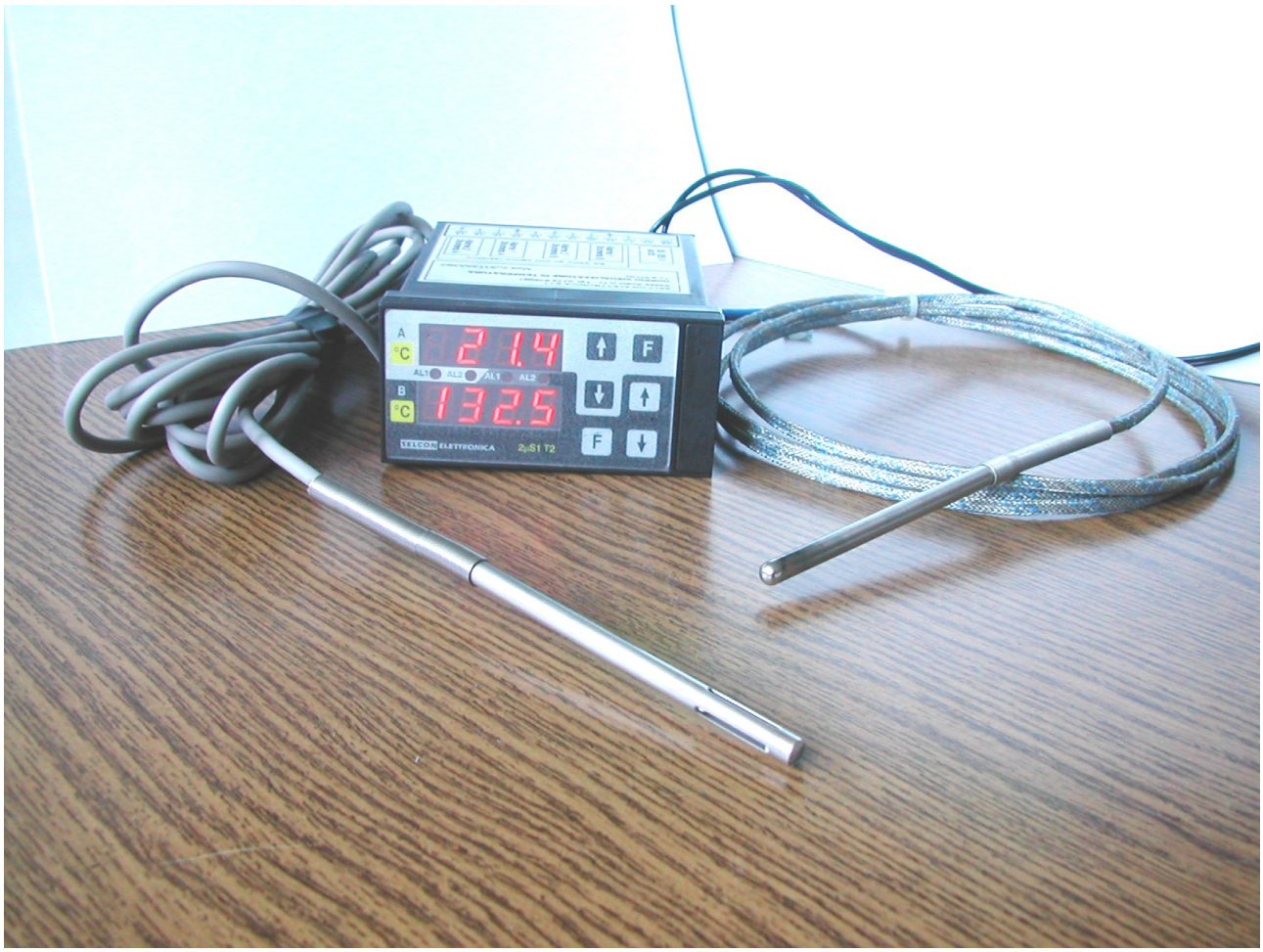
CONTENUTI:

1-Introduzione

Quando ho iniziato lo stage, l'obiettivo, che mi sono posto, è stato quello di avvicinarmi al mondo lavorativo, intraprendendo una esperienza concreta con scadenze da rispettare.
In ambito universitario, quasi sempre per ragioni di tempo, non si analizzano, a fondo, gli aspetti che sono, però, indispensabili nel mondo lavorativo perché, al di la del progetto su carta e della teoria che sta alla base del fenomeno che stiamo studiando, c'è uno strumento da realizzare, strumento che dovrà operare misure attendibili e funzionali a progetti in ambienti lavorativi diversificati.
Sapevo, per il buon esito del progetto, che bisognava rispettare dei tempi e che tutto il progetto dovesse essere corretto, robusto ed efficiente affinché si potessero ottenere misure precise.
Da qui ne conseguiva una responsabilizzazione individuale nel portare avanti le fasi del progetto.
Nell'affrontare lo stage, oltre agli obiettivi prefissati, ci sono delle conoscenze che sono al di fuori degli obiettivi che muovono il progetto ma che sono di contorno allo stage, e proprie della formazione del carattere lavorativo dell'individuo e del suo bagaglio culturale.
Gli stessi rapporti sociali, all'interno di quella piccola "famiglia" chiamata azienda, sono diversi da quelli da me conosciuti nelle altre esperienze di vita, compresa quella universitaria.
L'ingegner Mauro Vincenzoni è stato assegnato al mio progetto ed è stato per me il polo referente per qualsiasi necessità, dal reperimento del materiale, a quei piccoli consigli dettati con grande esperienza che permettono il lineare proseguimento dei lavori.
L'ingegner Vincenzoni è stato un punto di sicurezza, una persona affabile e sempre disponibile che non esitava a pungolarmi nel momento in cui facevo qualcosa che non rispondeva fedelmente al progetto.
Con gli altri addetti al progetto, siccome avevamo dei compiti ben definiti, i punti di contatto non erano molti quindi si è instaurato un rapporto più superficiale ma pieno di disponibilità e cortesia.
In un clima di disponibilità l'esecuzione dei lavori non poteva che svolgersi in modo naturale e positivo. Questo è forse quello che qualsiasi imprenditore dovrebbe riuscire a creare all'interno del suo nucleo lavorativo.
L'altro aspetto formativo di questo stage è il fatto che quasi tutti i manuali che ho dovuto consultare e studiare erano in inglese da cui consegue una maggiore dimestichezza con l'inglese tecnico; è stata di fondamentale importanza l'esperienza di aver affrontato lo studio di diversi testi universitari in lingua inglese.

2-Obiettivi dello stage

Sono stato associato al progetto di uno strumento elettronico digitale di utilizzo industriale:



Lo strumento tramite un sensore rileverà una grandezza fisica trasformandola in un segnale elettrico. Il segnale verrà acquisito da un microcontrollore che provvederà a linearizzarlo ed elaborarlo per poi inviarlo a un display che consentirà all'utilizzatore di leggere la misura di grandezza in questione.
Nel mio caso particolare la grandezza da misurare è una temperatura e questo verrà fatto con un trasduttore apposito di temperatura.Questo sensore si chiama Pt100, se ne parlerà nella sezione apposita. Nella specifica dello strumento abbiamo deciso di non considerare le temperature negative, perché il campo di applicazione del prodotto non necessita di queste temperature. Lo strumento verrà usato, infatti, per scopi diversi riguardanti temperature positive: misurare la temperatura raggiunta dagli avvolgimenti di un motore elettrico e attivare una ventola di raffreddamento qualora la temperatura superasse i valori limite, misurare la temperatura in forni per la sterilizzazione dei contenitori a uso farmaceutico, misurare la temperatura in apparati di frenatura come ruote di autocarri per prevenzione inneschi incendi.
La considerazione di tutte le temperature comprese in un range diverso da quello che stiamo considerando noi da 0°C a 660°C non comporta, di fatto, un aumento della complessità del problema ma solo una piccola modifica dei parametri.
La parte che mi riguarda del progetto è lo sviluppo software e, nella fase di messa a punto dello

strumento, cioè della combinazione tra hardware e software, interagirò direttamente con il resto del gruppo addetto all'hardware.
Tramite un circuito di acquisizione della informazione della temperatura, del quale fa anche parte la PT100 e un convertitore analogico digitale, si ottiene un numero esadecimale.
A questo numero è associata una e una sola temperatura che a questo punto può essere elaborata da un programma che visualizza sul display una delle temperature che opzionalmente l'utilizzatore può scegliere.
Infatti, sul frontale, ci sono tre pulsanti che a seconda dell'esigenza fanno visualizzare tre informazioni differenti.
Ciclicamente lo strumento fa la misura della temperatura.
Il primo pulsante visualizza sul display la massima temperatura misurata dal momento del reset.
Il secondo pulsante visualizza sul display la minima temperatura misurata dal momento del reset.
Il terzo pulsante resetta lo strumento ed indica, quindi, l'inizio dei tempi per quel gruppo di misure che l'utilizzatore sta facendo.
Il microcontrollore utilizzato è l'8051 e viene programmato in C tramite un programma che adatta leggermente lo standard C ANSI per la programmazione del microcontrollore.

3-Azienda ospitante

L'azienda sede del progetto è la SELCON elettronica S.r.l. situata tra Latina e Sezze Scalo sulla s.s.156 al chilometro 42,100.
E' una società che opera dal 1989 nel settore dell'elettronica per applicazioni industriali, composta attualmente da una quindicina di impiegati, ingegneri e periti in elettronica, informatica e telecomunicazioni . In particolare ha realizzato e produce di serie strumenti digitali per la misura di grandezze elettriche, della temperatura, del tempo ecc. nonché strumenti per la termoregolazione e per il conteggio degli impulsi.
Parallelamente a questa attività, la società ha sviluppato un settore applicativo in grado di studiare e realizzare apparecchiature elettroniche particolari (per esempio apparecchiature strumentali per misure e collaudi) dedicate alle esigenze specifiche del cliente e di proporle sia come unici esemplari sia come prodotti destinati ad una produzione di serie. Il progetto da me affrontato fa parte di questo settore dell'azienda nell'ambito della misura.
La Selcon intrattiene rapporti commerciali con aziende che operano nel settore della costruzione di macchine per la produzione, forni termoregolati, macchine per il packaging (impacchettamento), quadri per impianti di automazione e produzione in genere, in cui sia necessario misurare e controllare grandezze.
Alcune delle aziende più rilevanti sono: Packservice, Pfizer, Menarini, Lepetit, Gambro, Rendial e altre.
Per maggiori informazioni si può visitare il sito www.selconelettronica.it oppure chiedere tramite e-mail all'indirizzo selcon@selconelettronica.it.

4-Realizzazione

4.1-Strumenti utilizzati

4.1.1-Turbo C++

Nei corsi universitari da me frequentati, il linguaggio più utilizzato è il C++ secondo lo standard ANSI che viene utilizzato per apprendere i costrutti e gli aspetti teorici dell'informatica.
Data la mia familiarità acquisita con il linguaggio C si è preferito fare un prototipo in questo linguaggio per analizzare, in un ambiente al quale io sono avvezzo, le problematiche concettuali del programma e non distogliere la nostra attenzione con le peculiarità di un nuovo linguaggio.
Per la realizzazione del prototipo mi è stata data la libertà di scegliere tra le varie versioni e io ho scelto il Turbo C++ della Borland.
Per integrare, poi, il prototipo con la versione definitiva del programma scritta in C non standard, sono state fatte delle modifiche e aggiunte delle funzioni.

4.1.2-Pt100

La sonda utilizzata è la Pt100, in figura, dove Pt sta per platino, perché è fatta di questo materiale, e 100 sta per 100 Ω che è la resistenza a 0 °C.

La Pt100 non è altro che una resistenza che varia rispetto alla temperatura R(T).
A –200 °C si ha una resistenza di 18,52 Ω, a 0 °C si ha 100 Ω e a 660 °C si ha 332,79 Ω. La

funzione R(T) individuata dalla Pt100 è non lineare il che darà luogo allo sviluppo di una parte del programma apposita e di una serie di accorgimenti in più.

4.1.3-ISOTECH PRT

Per ottenere, dato un valore di temperatura o di resistenza, il corrispondente valore di resistenza o di temperatura, si utilizza un programma della ISOTECH chiamato PRT, dove P sta per Pt100, R sta per resistenza e T per temperatura.
E' possibile passare da temperatura a equivalente resistenza e viceversa ed il programma è user friendly, facile da usare.
L'ulteriore potenzialità del programma è che si può fissare un passo X di temperatura o di resistenza e ottenere tutti gli equivalenti valori, rispettivamente di resistenza o di temperatura dati due valori, iniziale e finale. Il valore iniziale verrà incrementato di X ogni volta e verrà calcolato il corrispondente valore.
Questo dà luogo alla creazione di una tabella che ci servirà per associare ad ogni valore un numero esadecimale N, dato il passo. Per ognuno di questi valori sono quindi legati due unici valori uno di resistenza e uno di temperatura.
In conclusione possiamo risalire alla temperatura tramite il numero esadecimale.

4.1.4-L'ADuC816

L'elevata scala di integrazione raggiunta dai moderni processi tecnologici permette attualmente di disporre di microcomputer integrati in un unico chip (single chip microcomputer).
Questi integrati dispongono al loro interno di un sistema completo, oltre che della CPU, di dispositivi di I/O, di una memoria interna RAM e di una memoria interna ROM o EPROM.
L'8051 è un single chip microcomputer progettato dalla Intel, disponibile in versione tecnologica HMOS e CMOS.
Le caratteristiche principali di questo microcomputer si possono riassumere in:

a) CPU ad 8 bit ottimizzata per le applicazioni di controllo;
b) capacità estesa di elaborazione booleana (su singolo bit);
c) 64 kbyte max di memoria indirizzabile per i programmi;
d) 64 kbyte max di memoria indirizzabile per i dati;
e) 4 kbyte di memoria per i programmi interna al chip;
f) 128 byte di memoria RAM per i dati interna al chip;
g) 32 linee di I/O bidirezionali indirizzabili separatamente;
h) 2 timer/contatori a 16 bit;
i) UART (Universal Asynchronous Receiver Transmitter) full duplex;
j) 5 linee di interrupt con due livelli di priorità;
k) generatore di clock interno al chip.

Lo schema a blocchi dell'architettura dell'8051 è mostrata in figura 1:

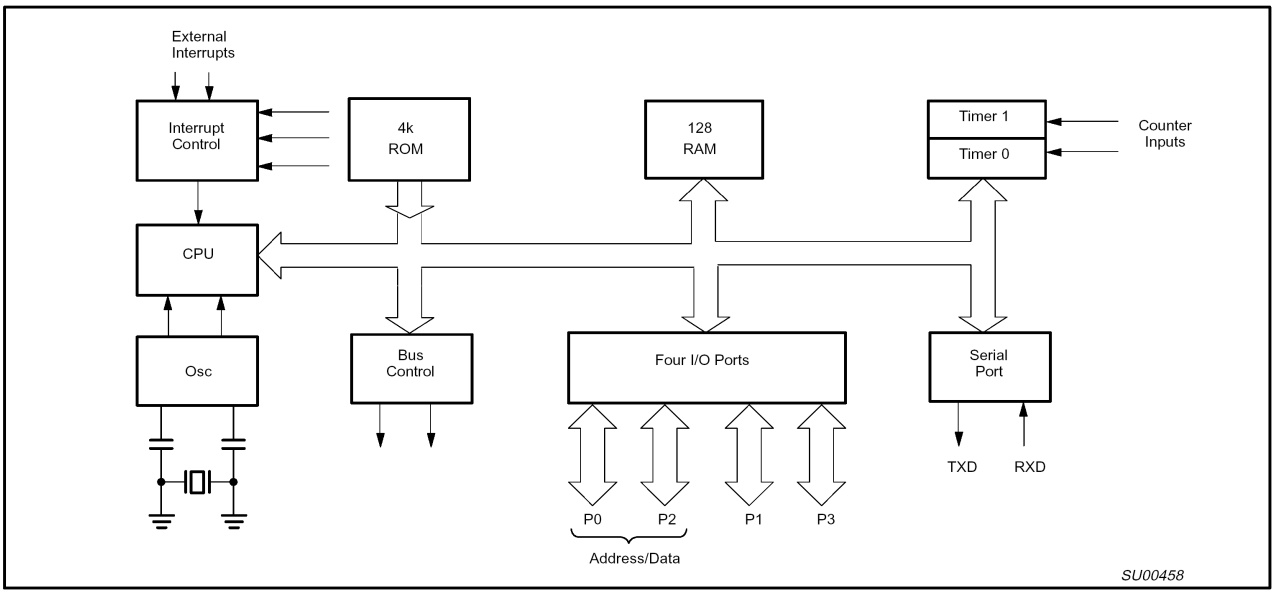


figura 1

L'integrato, che stiamo utilizzando, ha come base l'8051 ma ha dei componenti aggiunti come 2 ADC, 2 generatori di corrente interni e altro. Oltre la completa compatibilità con l'8051 e il suo successore l'8052, questo integrato contiene già, al suo interno, dei componenti fondamentali per il nostro scopo il che ci permette di ottimizzare il nostro problema in termini di: spazio, numero di componenti quindi affidabilità, minore onere per gestire ordini di magazzino, costo sviluppo schede e montaggio, manutenzione più semplice; tutte risorse fondamentali del progetto (basti pensare che per ridimensionare lo strumento di pochi centimetri ci vogliono alcune migliaia di euro!). Nella figura 2 la parte tratteggiata del core è quella dell'8051 mentre il resto è proprio dell'ADuC816.
In allegato c'è lo schema dettagliato dell'architettura interna dell'8051.
Osservando le figure 1, 2 e l'allegato si nota che, a parte alcuni registri specifici, si possono riconoscere le unità essenziali di un sistema a microprocessore e cioè ALU, registri fondamentali (program counter, accumulatore, stack pointer, registro di stato), RAM, ROM, I/O.
L'8051 ha memoria per i programmi (a sola lettura ROM o EPROM) e memoria per i dati (lettura/scrittura, dunque RAM) separate. Dal momento che la sua capacità di indirizzamento è di 64 k per ciascun tipo di memoria, è prevista una serie di terminali atti a pilotare una memoria esterna sia per i dati sia per i programmi, che può essere aggiunta a quella già presente internamente. L'ADuC816, ha come scelta particolare che lo caratterizza in parte, l'utilizzo di memorie Flash/EE per i dati e per i programmi. La memoria Flash/EE è una memoria non volatile e riprogrammabile che viene chiamata in questo modo per la sua velocità e perché sostanzialmente è una EEPROM. Come si vede dal diagramma funzionale che segue la memoria programmi è di 8kbyte, la memoria dati è di 640 byte e la RAM è di 256 byte.
La piedinatura dell'ADuC816 è disponibile come allegato.
Si rimandano ai data sheet le peculiarità del dispositivo per non uscire troppo al di fuori del compito prefissato dallo stage.

FUNCTIONAL BLOCK DIAGRAM

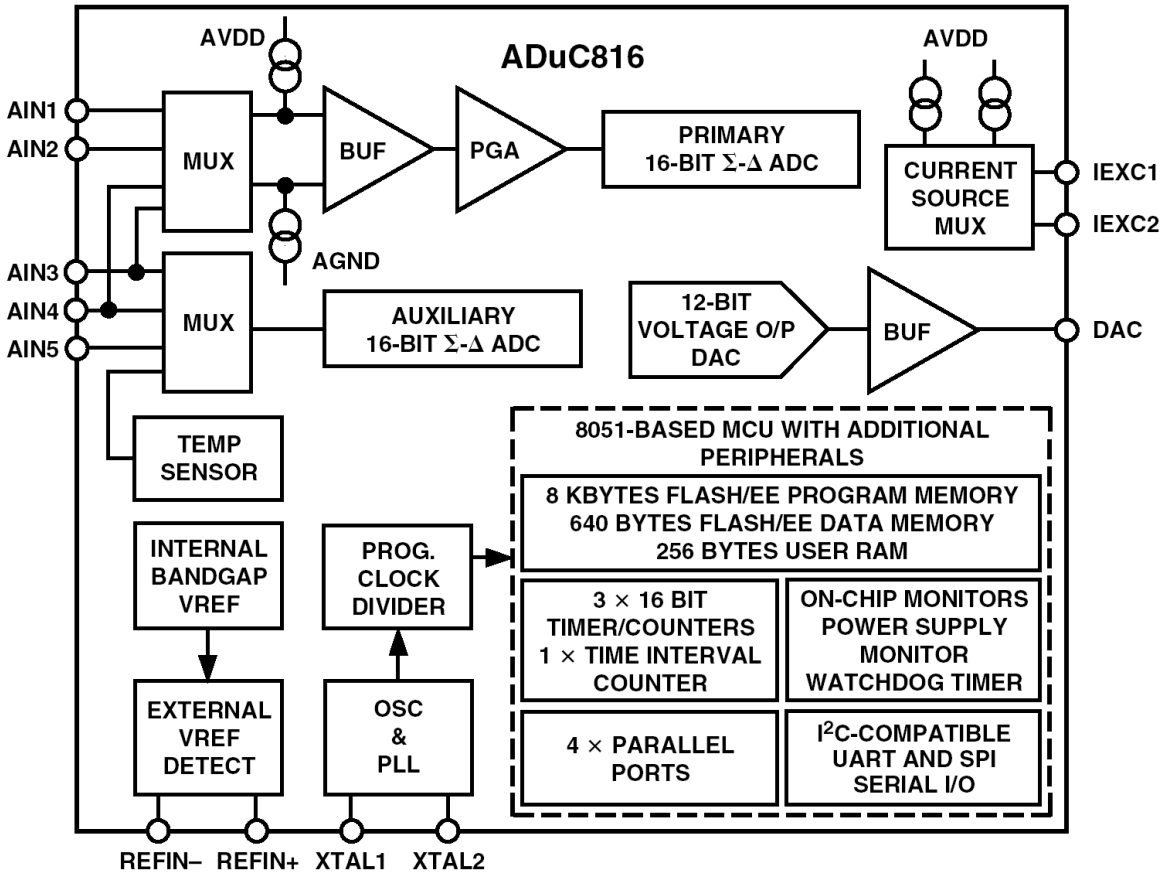


figura 2

4.1.5-IAR Systems

Il programma che viene utilizzato per sviluppare il codice si chiama IAR Systems Embedded Workbench. Stiamo parlando di un potente ambiente di lavoro integrato, per lo sviluppo di applicazioni in C o Assembler, per la famiglia di microprocessori 8051. Insieme all'Embedded Workbench è associato un debugger chiamato IAR C-SPY utile per il debugging del programma.
L'Embedded Workbench è molto simile al C ANSI ma aggiunge a questo una serie di istruzioni e variabili ambiente che ci permettono di pilotare da software i singoli piedini dell'ADuC816 e le altre risorse disponibili. La facilità con cui si maneggia l'ADuC816 è dovuta all'inclusione di una libreria ADuC816.h che permette al compilatore di riconoscere le istruzioni e i parametri ambiente su detti. Un linguaggio ad alto livello, quindi, ma che permette di sfruttare le potenzialità del processore direttamente, senza nessun intermediario o, come si usa dire, a basso livello.
Quando si utilizza questo programma bisogna stare attenti ai tipi utilizzati perché qualche volta questi sono ridefiniti o comunque perché sono stati aggiunti dei tipi che risultano comodi nella stesura del programma.
Il programma permette anche una elastica gestione di processi in background, tramite interrupt per esempio. Abbiamo utilizzato una istruzione che fa eseguire una routine di gestione dell'interrupt ogni volta che se ne genera uno cioè ogni 2 millisecondi.
Una volta scritto il programma, con la funzione build, generiamo il file eseguibile che per questo programma ha estensione .hex. Questo è il file che andrà caricato direttamente nell'8051 per programmarlo.
Mi sono chiesto molte volte come delle linee di codice, che sembrano una cosa tanto astratta, potessero poi pilotare un chip,che invece è una cosa tutt'altro che astratta, in questo caso questa mansione è svolta interamente dall'ambiente integrato.

4.1.6-interfaccia seriale

Si tratta di un cavo, cioè una connessione fisica, e di una piccola scheda, che servono per gestire la comunicazione tra il computer e lo strumento. Una volta sviluppato il software con il programma della IAR bisogna programmare l'8051 e per fare questo bisogna utilizzare una interfaccia hardware, cioè l'interfaccia seriale, e una interfaccia software, cioè il programma WSD, che gestisce la comunicazione. La trasmissione dati si effettua tramite porta seriale RS232. Allo strumento l'interfaccia seriale viene collegata tramite un connettore appositamente previsto nel progetto.

4.1.7-WSD

E' un programma della Analog devices con una interfaccia utente molto semplice che serve per programmare ed eventualmente far eseguire il programma caricato. Nel fare questo il programma chiede di selezionare il file eseguibile .hex generato dal programma della IAR.

4.1.8-2μS1T2

Il prodotto finale è il 2μS1T2.

La sigla è una composizione di lettere ognuna delle quali ha un significato ben preciso. Il 2 sta per doppio perché lo strumento permette di fare 2 misure di temperatura tramite 2 sonde (è una scelta industriale). Il μ ci dice che è un sistema a microprocessore. La S sta per Selcon, la ditta produttrice. L'1 indica che è la prima edizione alla quale sicuramente ne seguiranno altre (come si vede anche da alcune scelte hardware riservate per usi futuri). La T sta per temperatura perché il segnale in analisi è di temperatura. Il 2 sta per 2 led, per ogni μS1T dei 2 inclusi nell'involucro, per un totale di 4 led riservati per usi futuri(come per esempio allarmi che si accendono quando la temperatura in questione è troppo alta). Il 2μS1T2 è formato dai seguenti componenti hardware:

1) Pt100;
2) Involucro contenitore di plastica con display;
3) Scheda alimentatore;
4) Scheda display;
5) Scheda CPU;

Le schede sono tutte collegate tra loro e ognuna svolge una funzione propria, vitale per il corretto funzionamento del dispositivo.

L' hardware è stato interamente progettato dall'ingegner Vincenzoni che, utilizzando le sue conoscenze su altri dispositivi radicalmente diversi da questo, è riuscito, come un puzzle, ad ottenere un nuovo strumento riuscendo non solo ad assemblare i diversi pezzi ma ottimizzando le varie scelte per il nostro particolare problema. La scelta dei componenti giusti non è difficile ma per scegliere i componenti migliori presenti sul mercato sono necessarie conoscenze appropriate. Questa fase ha grande importanza per la futura vita dello strumento perché una maggiore qualità renderà sicuramente più competitivo sul mercato lo strumento.

La scheda di alimentazione è stata interamente rifatta da Vincenzoni ottimizzando il tutto per lo scopo prefissato, in questo caso minor consumo.

La scheda CPU è quella che contiene i componenti più importanti dello strumento.

Una prima parte è quella della acquisizione del segnale tramite la sonda Pt100 e della trasformazione di questo in un segnale digitale N, figura 4:

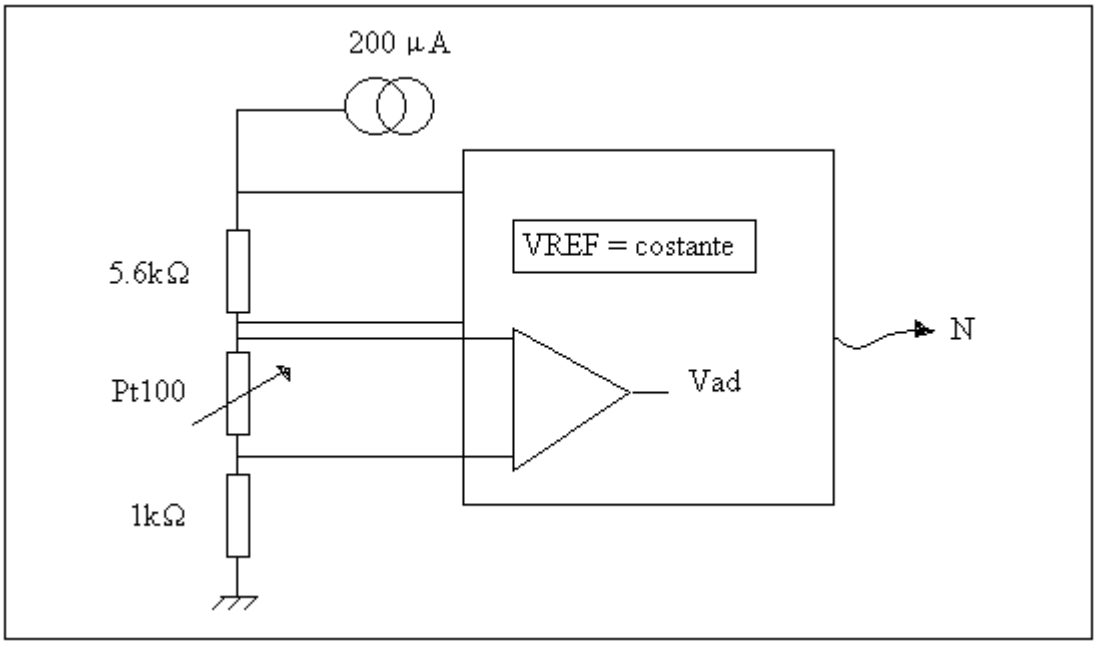


figura 4

Tramite un generatore di corrente si fa scorrere nel circuito una corrente (fissa) di 200 µA che passerà per una resistenza fissa 5,6 kΩ ai capi della quale ci sarà, quindi, una tensione costante. La tensione costante viene usata come riferimento nella conversione analogico/digitale VREF, della quale il convertitore ha bisogno per funzionare.
La corrente a questo punto passa sulla resistenza variabile, che è la Pt100, ai capi della quale ci sarà, quindi, una tensione variabile rispetto alla resistenza, cioè alla temperatura (guardare la parte sulla Pt100). Il convertitore confronterà la tensione variabile con la tensione di riferimento VREF e darà come risultato un numero esadecimale N collegato alla temperatura misurata dalla sonda secondo il procedimento appena descritto. L'ultima resistenza, quella collegata con la massa, serve solo per regolare il range di valori di tensione che il convertitore può accettare.
Questo è il principio sul quale è stato costruito il circuito, ma in realtà c'è da aggiungere qualche altra considerazione di carattere pratico. Per piccole variazioni della resistenza aumenta di molto la temperatura, bisogna tener conto anche della resistenza introdotta dai fili di collegamento della sonda allo strumento.
I tre fili, quelli ai capi del convertitore e quello diretto a massa, introducono tre resistenze uguali e tramite un generatore uguale all'altro, cioè di 200 µA, si può compensare l'effetto indesiderato del filo. Il generatore va inserito sul secondo filo collegato al convertitore e, se si analizzano i versi delle correnti e i segni ai capi delle resistenze, si vede che gli effetti si eguagliano rendendo ininfluenti le resistenze in più. Il secondo generatore, come il primo, è incluso nell'ADuC816.
Nel circuito intero della scheda, figura 5, viene aggiunto il microprocessore e alcuni altri circuiti collegati ad esso per noi meno interessanti: quarzo, alimentazione, filtri passa basso e altri dispositivi per ripulire il segnale dal rumore, ecc….

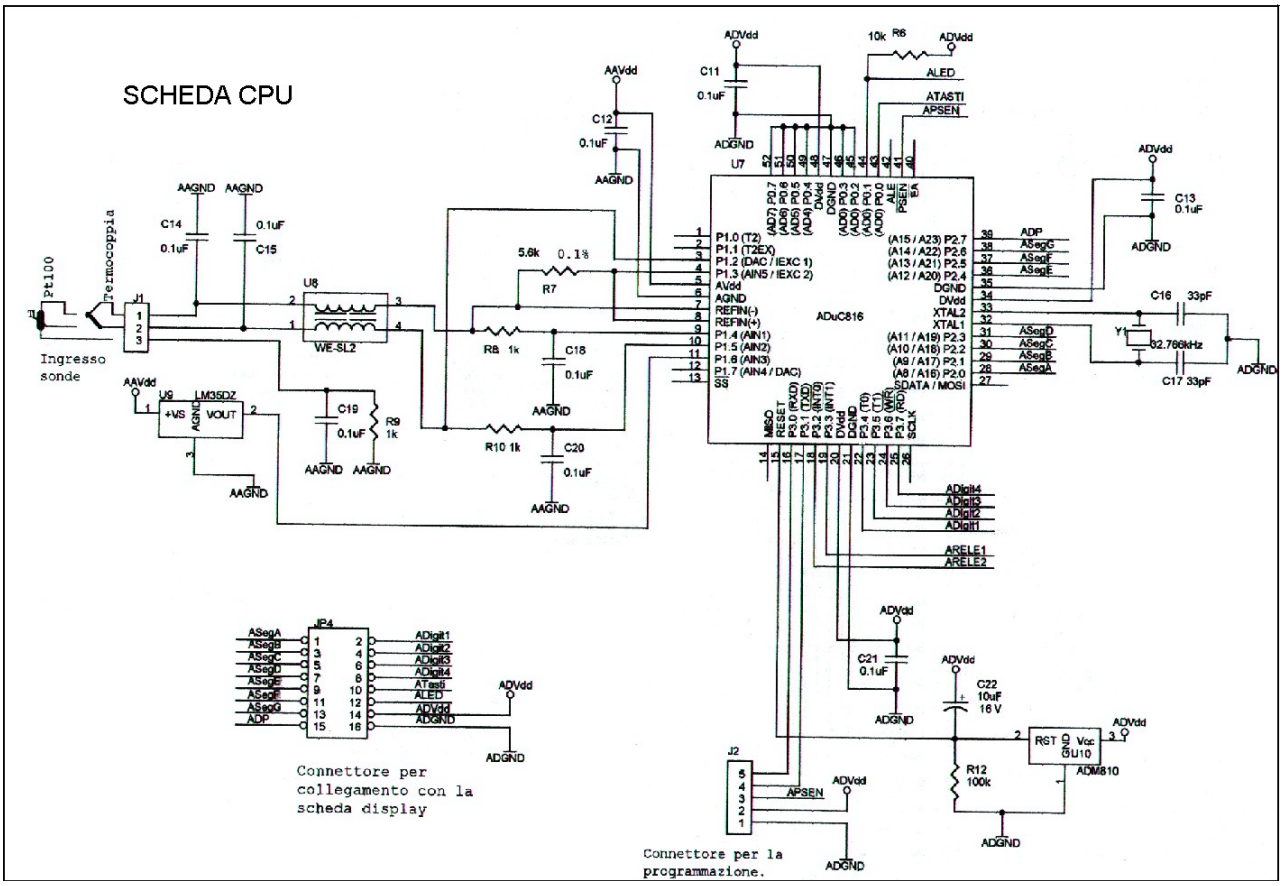


figura 5

Nella parte a sinistra del processore si riconosce il circuito di base per l'acquisizione con l'aggiunta di filtri e altri dispositivi per ripulire dal rumore.
Questa parte del circuito va collegata ai pin appropriati del microprocessore 8051 che conducono all'ADC e alla VREF (REFIN(-) e REFIN(+)).
Per capire a fondo il componente principale rivolgersi alla parte appropriata e ai data sheet.
La scheda display permette la visualizzazione dell'elaborato, figura 7.
Per risparmiare dei collegamenti che renderebbero più caotica questa scheda si utilizza un accorgimento. Quando un diodo led, che era acceso, viene spento, intercorre una frazione di secondo prima che l'occhio umano veda il diodo led spento. Si può pensare quindi di gestire la scheda in modo che un digit alla volta viene controllato dal circuito mentre gli altri per quel fenomeno ottico descritto sono ancora accesi per "inerzia", a patto che il circuito sia abbastanza svelto a passare in sequenza da un digit all'altro.
Questa politica si chiama multiplexing cioè linee fisiche comuni vengono multiplexate tra i 4 digit con un notevole risparmio di collegamenti.

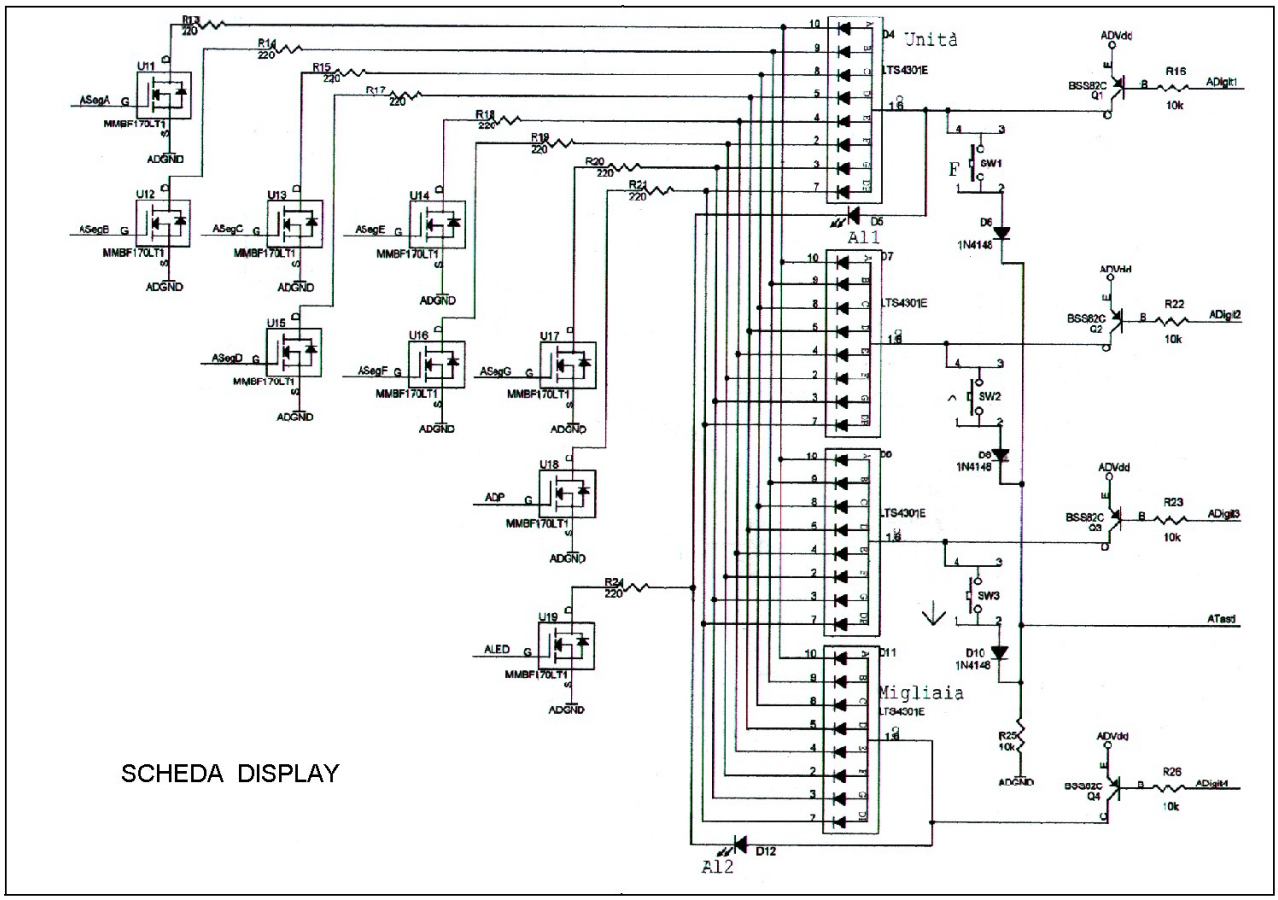


figura 6

Si utilizzano digit a 7 segmenti che sono composti da 8 diodi led; i primi 7 sono i segmenti identificati da a fino a g e l'ottavo è il punto identificato con dp, figura 7.

Per accendere il singolo segmento bisogna mettere a 1 il relativo bit del numero a 8 bit (uno per ogni segmento che fa parte del digit) che identifica quel particolare led. I segmenti sono così associati:

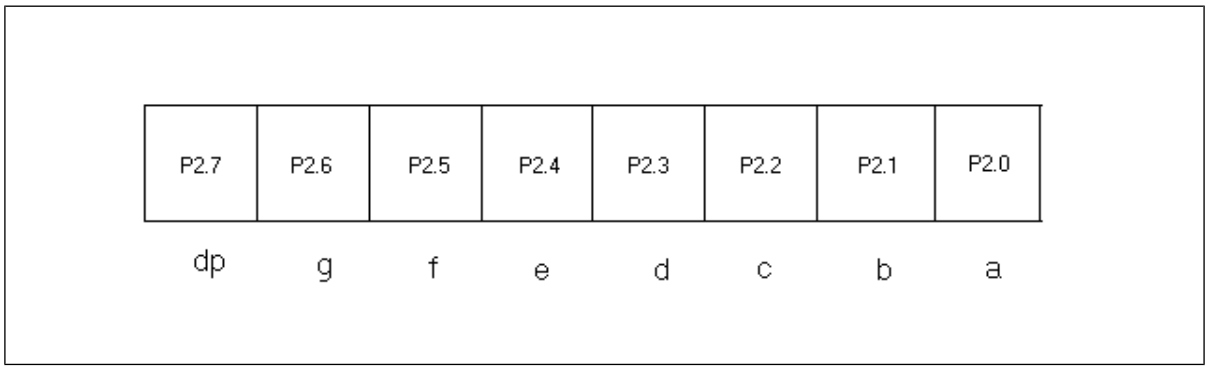
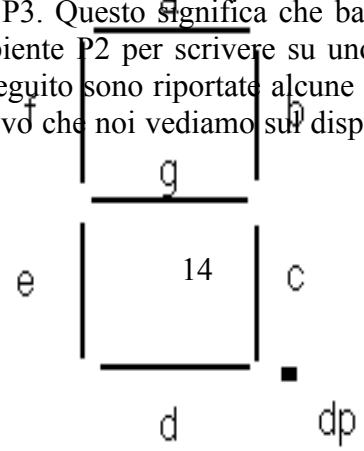


figura 7

La variabile ambiente P2 conterrà il numero a 8 bit che verrà poi visualizzato sul display. Il digit, invece, si seleziona settando uno dei bit del parametro ambiente P3. Questo significa che basterà inizializzare, al valore che si vuole visualizzare, la variabile ambiente P2 per scrivere su uno dei digit sul display selezionato dall'appropriato settaggio di P3. Di seguito sono riportate alcune delle codifiche che associano al numero esadecimale P il simbolo effettivo che noi vediamo sul display a sette segmenti di figura.



0x3F = "0" 0x77 = "A"

0x06 = “1”	0x00 = “ “ = BLANK
0x5B =”2”	0x1F = “b“
0x4F = “3”	0x0D = “c“
0x66 =”4”	0x39 = “C“
0x6D =”5”	0x5E = “d”
0x7D =”6”	0x79 = “E”
0x07 =”7”	0x47 = “F”
0x7F =”8”	0x5E = ”G”
0x67 =”9”	0x76 = “H”

Sul display ci sono anche altri diodi led (AL1 e AL2) previsti dall’hardware per possibili utilizzi futuri. Per avere una idea del display definitivo dello strumento possiamo guardare la seguente foto:

La gestione del circuito fa largo uso di transistor mosfet e, per quanto riguarda i diodi, la connessione tra loro scelta è quella a catodo comune.
Tralascio gli altri dettagli per non addentrarci troppo nell’hardware.

4.2-Fasi e problematiche

4.2.1-Linearizzazione

A partire dalla resistenza,scelto un determinato passo, seguendo dei criteri, si genera una tabella con PRT associando, quindi, a quegli intervalli regolari della resistenza i relativi valori di temperatura. Il criterio che si sceglie è quello di generare 2^{16} valori tra T=-200 °C e T=660°C.
Quello che noi vogliamo è associare ad un numero esadecimale N a 16 bit, biunivocamente, un numero reale di temperatura. Per questo scegliamo 2^{16} valori perché permette di associare valori esadecimali da 0000h a FFFFh essendo il numero N, in uscita dal convertitore, a 16 bit.
Notare che la caratteristica R/T della Pt100 è non lineare.
Per associare N a R e T basta aggiungere un'altra colonna alla tabella, questa colonna conterrà valori crescenti di N ai quali, in questo modo, i valori di T risultano associati direttamente. Quando, scrivendo il software, si ha bisogno, dato un N, di conoscere la relativa temperatura, basta accedere a una tabella con tre colonne e cercare per corrispondenza i valori che ci interessano.
Sarebbe tuttavia ingenuo memorizzare una tabella di tre colonne quando il legame che ci interessa, a questo punto, è N/T. Scegliamo di memorizzare una tabella a due colonne N e T dove la caratteristica N/T è ancora non lineare.
Da ricordare che nelle specifiche dello strumento consideriamo solo le temperature positive.
Il nostro range di valori di N per i quali la temperatura è positiva va da 8000h a FFFFh che equivale a 2^{15} valori. Questo è un numero molto grande e per noi una delle risorse più importanti, perché scarsa è la memoria. Dobbiamo cercare di ridurre al minimo la tabella e, soprattutto, evitare di memorizzare informazioni ridondanti.
Esiste una tecnica che ci permette di memorizzare molti meno di 2^{15} elementi con un margine di errore impercettibile. Si sceglie un passo esadecimale X e si prendono dalla tabella N/T solo gli n elementi dati da 8000 + n*X per ogni n naturale fino a che non si raggiunge il nostro limite superiore FFFFh.
La struttura della tabella è:

N	T
8000	0
8000 + x	T1
8000 + 2x	T2
8000 + 3x	T3
-	-
-	-
-	-
8000 + (n-2)x	T_{n-2}
8000 + (n-1)x	T_{n-1}
FFFF	660

N = esadecimale
T = gradi centigradi
x = esadecimale

Nel caso ci servisse di sapere quanto vale la temperatura in corrispondenza di uno degli N scartati, diciamo Nx, possiamo fare una approssimazione di linearità tra il valore precedente e il valore successivo di N e T. Questa approssimazione è generalmente chiamata interpolazione lineare. Tanto più la caratteristica N/T è non lineare maggiore sarà l'errore sull'approssimazione fatta, tramite l'interpolazione, sulla temperatura. Nel nostro caso l'approssimazione è accettabile perché la temperatura richiesta si vuole con la precisione del decimo di grado.
La formula di interpolazione è data dalla retta che passa per due punti (Ni,Ti) e (Nf,Tf) :

$$Tx = Ti + \frac{Tf - Ti}{Nf - Ni} \cdot (Nx - Ni)$$

dove:
Tx è il valore di temperatura corrispondente ad un valore N non memorizzato nella tabella;
Ti e Tf sono i valori di temperatura corrispondenti al valore Nf (N precedente a all'Nx dato) e Ni (N successivo all'Nx dato).
ESEMPIO di interpolazione lineare:

N	T
8000	0
9000	20
-	-
-	-
-	-

N = esadecimale
T = gradi centigradi

Qual'è la temperatura per N = 8026h?

Dai dati abbiamo:
Ni = 8000h con Ti = 0°C
Nx = 8026h → Tx = ?
Nf = 9000h con Tf = 20°C
A questo punto basta sostituire i dati alla formula di interpolazione per ottenere la temperatura cercata.
In questo modo abbiamo, quindi, diminuito drasticamente la memoria occupata utilizzando una tabella che ha le dimensioni più modeste di n+1 righe e 2 colonne.
Ragionando sulla tabella si vede subito che sarebbe ridondante memorizzare entrambe le colonne. L'informazione indispensabile per noi è quella della temperatura perché N può essere ricostruito tramite l'indice i della tabella, il passo X e il primo valore (8000h nel nostro caso).
La tabella che inizialmente aveva tre colonne e 2^{15} righe, nel programma in C diventa un array di reali lungo n+1, più la costante che serve da indice per ricostruire la informazione della N.

4.2.2-Prototipo base

Realizzazione di un programma che, a partire da un numero esadecimale a 4 cifre Nx, ci permette di ottenere la temperatura Tx tramite la formula di interpolazione dove gli elementi che ci servono sono memorizzati in una tabella del tipo visto al paragrafo precedente.
Abbiamo bisogno di alcune funzionalità:

1) Dato Nx accedi alla tabella e ritorna il valore memorizzato Ni immediatamente precedente a Nx.
   - esadecimale N_precedente(esadecimale Nx);
2) Dato N ritorna la temperatura, in tabella, corrispondente a quel particolare N.
   - reale T_relativo(esadecimale N);
3) Dato Nx calcola Tx.
   - reale calcola_Tx(esadecimale Nx);

Descrizione delle funzioni C:

1) esadecimale N_precedente(esadecimale Nx):
   inizializiamo una variabile N a 8000h e la facciamo ciclare con un while, aggiungendo ogni volta il passo X, fino a quando l'N è minore o uguale a Nx. A questo punto per ottenere l'N desiderato basta sottrarre X a N.

2) float T_relativo(int N):
sottraiamo 8000h a N e calcoliamo l'indice della tabella che mi fa ottenere la temperatura desiderata. Questo è dato dal nuovo N diviso per X. Ritorniamo il valore della tabella calcolato per quell'indice.

3) float calcola_Tx(int Nx):
per calcolare Tx bisogna applicare la formula di interpolazione: calcoliamo Ni, Ti, Tf e sostituiamo nella formula notando che Nf – Ni = X quindi è inutile calcolare Nf, e che Tf e Ti si calcolano passando, alla funzione T_relativo, i parametri Ni + X e Ni.

Per risolvere il nostro primo problema è sufficiente implementare queste tre funzionalità passando facilmente da questa spiegazione descrittiva al codice C. Il programma si può strutturare in diversi modi ma bisogna cercare di utilizzare il minor numero possibile di istruzioni, costanti e funzioni perché la risorsa importante per noi è la memoria.
Si poteva semplificare il tutto prevedendo delle funzioni T_precedente, T_successivo e N_successivo ed eliminare T_relativo in modo che l'unico parametro che dovevamo dare in input a qualsiasi funzione era sempre Nx ma in questo modo due funzioni diventano superflue.

Integriamo ora il programma con le altre informazioni importanti quali la generazione della tabella, la scelta del passo e altre considerazioni oltre a quelle già fatte.
Prima di tutto notiamo che, la piccola complicazione di considerare N esadecimale, non esiste perché comunque il C convertirà questo numero, una volta inizializzato, come un normale numero decimale. Nella versione definitiva del programma utilizzeremo questo accorgimento.
Il numero di righe dell'array che dobbiamo memorizzare può essere compreso tra 50 e 150 vista la disponibilità in memoria. Dobbiamo prendere X in modo tale che il numero di elementi sia compreso in quell'intervallo e che sia un multiplo di 2 (perché in questo modo X verrà intero). I numeri che rientrano in queste ipotesi, sono 64 e 128.
Scegliendo di memorizzare $64 = 2^6$ elementi, il passo X viene calcolato distribuendo sui 2^{15} valori di temperatura (da 8000h a FFFFh), i 64 elementi. Dividendo questi due numeri otteniamo
X = 512 , numero di cui sapevamo a priori il fatto che era intero perché dato dalla divisione di potenze di due.
A partire dal numero X mi serve di sapere il passo della resistenza Xr da dare in ingresso al programma PRT per generare la tabella che verrà, poi, utilizzata dal programma. Tra N e R c'è linearità cioè una retta identificata dalla formula:

$$R = Ri + \frac{Rf - Ri}{Nf - Ni} \cdot (Nx - Ni)$$

$$\frac{Rf - Ri}{Nf - Ni} = \text{coefficiente angolare}$$

Per il calcolo del passo Xr dobbiamo sostituire i valori:

Ri = R(0°C) = 100 Ohm
Rf = R(660°C) =330 Ohm
Ni = 8000h
Nf = FFFFh
Nx =8000h + (512)h = successivo di 8000h

Il passo Xr è dato da R – Ri ed è uguale a 3,637373438 Ω.

Per poter utilizzare il programma PRT abbiamo bisogno del passo, la resistenza iniziale 100 Ω e la resistenza finale 330 Ω. Una volta generata la tabella viene eliminata la parte relativa alle resistenza e, tramite un semplicissimo taglia e incolla, copiata all'inizio del programma per essere assegnata ad un array C.
Oltre alle funzionalità del programma precedente il prototipo base dovrà gestire i tre pulsanti situati sulla mascherina. Abbiamo già detto che il primo e il secondo pulsante visualizzano il massimo e il minimo delle temperature che sono state calcolate dal reset. Il terzo pulsante resetta lo strumento. Se nessun pulsante viene premuto, sul display viene visualizzata la temperatura dell'ultima misura. Questa situazione viene gestita inserendo tre variabili boleane, una per ogni tasto. Se il tasto è premuto, la variabile boleana corrispondente è settata. Nel prototipo siamo noi a settare le variabili per simulare che un pulsante viene pigiato ma nel programma definitivo ci sarà un processo in background che setterà automaticamente le variabili.
Lo strumento da quando viene messo in funzione leggerà continuamente, ogni dato tempo, i segnali di temperatura dall'esterno che verranno convertiti, come già visto, in tanti numeri esadecimali. Per ognuna di queste misure, bisogna controllare le variabili boleane per vedere se un pulsante è stato premuto. Il prototipo avrà, dopo la dichiarazione e inizializzazione di variabili tra cui la temperatura massima e minima misurata dal reset, un ciclo infinito con una serie di controlli su cosa deve essere visualizzato, cioè sui pulsanti, e per ogni ciclo una conversione da N a T secondo il programma precedente.
Il codice C dei prototipi è allegato nell'appendice.

4.2.3-Visualizzazione sul display e timer

Nei prototipi base abbiamo simulato il funzionamento dello strumento interagendo direttamente con il monitor tramite librerie standard del C (conio.h e stdio.h) che ci permettono di gestire l'acquisizione e la stampa su dispositivi standard (tastiera e monitor) e altre piccole funzionalità irrilevanti per il nostro programma. Il programma in questo modo diventa molto grande perché il preprocessore sostituisce interamente, agli header file, il codice completo delle librerie. Questo non deve preoccuparci perché il punto che adesso svilupperemo serve per passare da una stampa a video a una su display, e da una acquisizione da tastiera a una acquisizione da Pt100.
Costruiamo un programma che ci permette di gestire le stampe sul display.
Per sviluppare questo programma abbiamo bisogno delle conoscenze hardware dello strumento, introdotte nella sessione 4.1.6.
Dallo schema della scheda CPU si vede che parte della porta P3 viene usata per abilitare uno dei 4 digit. Gli 8 bit della porta P2 vengono utilizzati per accendere o spegnere i diodi led che compongono il digit. Tramite il linguaggio C della IAR, si può accedere direttamente ai singoli pin del microprocessore tramite l'utilizzo di alcune variabili ambiente.
Se si vuole accendere il digit 1 bisogna settare il pin 22 tramite il comando P3.4=1 dove P3.4 è una variabile ambiente della IAR systems e, settandola a 1, selezioniamo il digit uno. P3.5, P3.6 e P3.7 identificano rispettivamente il 2°, il 3° e il 4° digit del display. L'istruzione software P2.0=1 accende il diodo led "a" del digit a 7 segmenti selezionato dalla porta 3 come si vede dal grafico della scheda CPU (vedere anche la parte relativa nel paragrafo 4.1.6).
Fatte queste considerazioni, possiamo scrivere una procedura "digit", che abilita in continuazione il digit successivo a quello che attualmente è selezionato e visualizza su questo il numero che si vuole visualizzare su quel digit.
Per fare questo, definiamo un array di 4 elementi, uno per ogni digit, che contiene i valori esadecimali rappresentanti il numero relativo dell'i-esimo digit. Per visualizzare il numero i-esimo della cifra che si vuole visualizzare sul display, basta copiare in P2 il valore i-esimo dell'array. La stampa sul display in questo modo equivale a scrivere nell'array.

Definiamo inoltre un contatore modulo 4 che identifica il digit del display che è attualmente in uso.
In questo modo la procedura "digit" assume una forma abbastanza semplice.
A seconda del valore della variabile contatore (tramite un case) si disabilita il digit corrente e si abilita il digit successivo, si scrive su P2 il valore dell'array relativo a quel valore del contatore e si aggiorna il valore del contatore in modo che la prossima volta che la procedura verrà eseguita si continui il ciclo dei digit.
Creiamo un file "visualizzazione" che gira in background. Questo programma include la procedura "digit" e l'inizializzazione del timer. Il timer genera l'interrupt che serve per gestire il passaggio da un digit all'altro. Nella sintassi dell'interrupt specifichiamo: il tipo di interrupt, la zona di memoria tra le 4 esistenti per operare il context switch e la routine di gestione. Il lasso di tempo che intercorre tra un interrupt e un altro viene deciso da noi mediante la memorizzazione di un valore all'interno di un registro specifico chiamato TH0. Scegliamo di generare un interrupt ogni 2 millisecondi. L'importanza nell'inizializzare bene il timer risiede nel fatto che a causa della nostra politica di multiplexing si rischia di avere per qualche piccolo istante, però percepibile dall'occhio umano, i digit spenti il che pregiudica il risultato finale del prodotto.
Nel file "visualizzazione" bisogna includere la libreria AduC816.h che fa riconoscere al nostro programma i parametri ambiente. Per inizializzare il timer bisogna settare una serie di registri il che equivale a impostare le funzionalità del timer a nostro piacimento.
Prendiamo le seguenti decisioni:

1) TR0 = 1 e TMOD = 0x12h; Si sceglie il timer 0, dei 2 timer a 8 bit dell'8051, con autoreload. Autoreload significa che dopo aver finito di contare il registro contatore TL0 viene automaticamente ricaricato al valore TH0 impostato da noi. TR0 è un bit che fa parte del registro TCON e che si può accedere direttamente.
2) ET0 = 1 e PT0 = 1; Si abilita l'interrupt del timer 0 e gli si assegna la priorità massima.
3) EA = 1; Si abilitano tutte le interruzioni.

Il file che conteneva il prototipo viene modificato eliminando tutte le printf e le scanf (o meglio tutto quello che riguarda l'interscambio di informazioni dell'utente con il monitor), dichiarando un array che contiene i valori dei 4 digit e inserendo la funzione imposta_array che prende in input un reale.
La funzione imposta_array serve, dato un numero reale r, a scomporre r nelle cifre che lo compongono e memorizzare queste all'interno di un array di 4 elementi. Il segnale di temperatura, infatti, è un numero reale che deve essere scritto sul display, cioè sui 4 digit. I digit avranno come risultato finale quello di visualizzare un numero reale r fatto in questo modo: r = XXX.X cioè con una cifra decimale o, meglio, con la risoluzione del decimo di grado. Questo significa che lo strumento è preciso al decimo di grado. Per scomporre il numero reale che identifica la temperatura nei numeri che saranno scritti sui digit abbiamo bisogno della funzione imposta_array.
In imposta_array tramite un meccanismo di moltiplicazioni e divisioni, intere e con resto, riusciamo a risolvere il problema che ci siamo proposti cioè smembrare il reale in 4 cifre distinte salvandole nell'array. Partendo dal reale r = XXX.X l'algoritmo è il seguente:
Moltiplichiamo r per 10 ottenendo r1 intero. Facciamo una divisione di r1 per 10 con resto, che sarà proprio il primo numero decimale che cercavamo. Facciamo una divisione intera per 10 di r1 ottenendo r2 che è un numero a 3 cifre intero. Il ciclo continua, facciamo una divisione di r2 per 10 con resto, che sarà il secondo numero decimale che cercavamo. Facciamo una divisione intera per 10 di r2 ottenendo r3 che è un numero a 2 cifre intero. Iterando altre due volte questo procedimento otteniamo separate tutte le 4 cifre.
In imposta_array è definito un'altro array di 10 elementi che contiene la conversione delle dieci cifre decimali, cioè il numero esadecimale corrispondente alla visualizzazione sul digit del numero decimale (guardare figure del paragrafo 4.1.6). Per ottenere il numero esadecimale che rappresenta

la conversione del numero decimale basta utilizzare il numero decimale come indice e accedere all'array di 10 elementi. Ogni volta che, nell'algoritmo precedente, ottenevamo la i-esima cifra desiderata, bastava utilizzarla da indice in questo array per ottenere il valore esadecimale che andava memorizzato nell'array di 4 elementi alla i-esima posizione.
Nel codice ci sono delle variabili dichiarate esterne perché sono dichiarate in file esterni. C'è anche la funzione esterna inizializza_timer che sta nel file visualizzazione. Inoltre si fa utilizzo del tipo bit non appartenente allo standard C ANSI ma definito dalla IAR systems.
Molte delle informazioni trattate in questo paragrafo, per l'inizializzazione del timer, sono state reperite dai data sheet.

4.2.4-Convertitore ADC

Come per il timer ci sono dei parametri ambiente relativi al convertitore per fare in modo che questo si comporti nel modo migliore per noi nei limiti imposti dall'8051. Tratterò l'interpretazione dei singoli bit in modo abbastanza superficiale perché questo va al di fuori degli obiettivi dello stage essendo, queste, delle scelte hardware. Il convertitore va prima inizializzato e poi gestito, implementeremo 2 funzioni per questi 2 compiti. Per capire meglio i numeri ai quali vengono inizializzati i registri bisogna fare riferimento ai data sheet.
Nella fase di inizializzazione del convertitore dobbiamo settare 4 parametri ambiente:
1) ADCMODE = 22h; riguarda il modo di funzionamento dell'ADC e viene impostato a 0010 0010 = 22h. La scelta di ogni bit ha un significato ben preciso per il convertitore. Ci sono 2 canali, noi ne utilizzeremo 1, quello primario e lasceremo inattivo l'ausiliario. Scegliamo inoltre di far funzionare l'ADC in single conversion mode invece che in modo continuo, scelta non importante ma che influenza il programma. Questo infatti significa che, per ogni nuovo Nx, bisogna settare di nuovo il registro ADCMODE a 22h.
2) ADC0CON = 43h; registro di controllo per il canale primario (0) dell'ADC viene impostato a 0100 0011 = 43h. Si sceglie la referenza esterna cioè la VREF viene impostata dall'esterno e non viene usata la referenza interna. Scegliamo l'organizzazione bipolare invece di quella unipolare.
3) ICON = 03h; registro di controllo per la corrente sorgente (gestisce le correnti).
4) SF = 255; stabilisce la velocità con cui lavora il convertitore. Settandolo in questo modo facciamo lavorare il convertitore nel modo più lento e più preciso possibile. Riesce a fare una conversione ogni 186,77 ms quindi 5 o 6 conversioni al secondo. Essendo il segnale una temperatura che cambia, quindi, molto lentamente va bene scegliere l'opzione più lenta.
Nella funzione "inizializza_convertitore" oltre al settaggio dei registri bisogna anche occuparsi della calibrazione dello strumento di cui si parlerà al paragrafo 4.2.5.

La fase di gestione deve utilizzare il bit RDY0 del registro ADCSTAT ovvero lo status register del convertitore. RDY0 ci indica, quando vale 1, che la conversione è stata effettuata. Nel programma di gestione bisognerà fare un test su questo bit se si vuole sapere quando è disponibile un nuovo Nx. La funzione si chiama "leggi_Nx" e si può implementare indifferentemente con 2 metodologie: interrupt e polling. Scegliendo di testare il bit RDY0 decidiamo di attuare una politica di tipo polling e, in questo caso il problema si modella molto facilmente in questo modo. Nell'altro modo si gestisce l'interrupt generato dall'ADC quando la conversione viene effettuata.
Nx è il numero in uscita al convertitore ed è dichiarato globalmente. La funzione legge da 2 registri a 8 bit: ADC0M e ADC0H. Tramite una maschera questi 2 registri vengono messi insieme nella variabile globale Nx che è a 16 bit. Nx viene cambiato ogni qual volta RDY0 diventa 1, cioè appena generato un nuovo Nx, e solo in quella occasione.
A questo punto bisogna modificare il programma in modo da richiamare le 2 funzioni nel modo corretto. La prima "inizializza_convertitore" deve essere messa fuori dal ciclo infinito, la seconda "leggi_Nx" invece dentro. Le funzioni sono allegate in appendice.

4.2.5-Calibrazione

L'ADuC816 mette a disposizione una memoria Flash/EE per i dati e una per i programmi. La memoria dati è divisa in 160 pagine ognuna contenente 4 byte. Per gestire la memoria Flash/EE l'AduC816 mette a disposizione 3 registri che noi settiamo secondo le nostre necessità.
In ECON mettiamo il tipo di operazione che vogliamo fare sulla memoria: lettura, scrittura, verifica, cancellazione.
EADRL contiene l'indirizzo della pagina di memoria sulla quale si vuole attuare il comando.
Le operazioni che si vogliono fare sulla memoria passano sempre per 4 registri. Se si vuole scrivere in una pagina, si scrive nei 4 registri quello che si vuole memorizzare, si imposta l'indirizzo in EADRL e si sceglie ECON = 02h che equivale a scrivere. Allo stesso modo per leggere una pagina all'indirizzo EADRL, scriviamo ECON = 01h e otteniamo il risultato della lettura nei 4 registri che si chiamano EDATA1—4.
Calibrare uno strumento significa farlo funzionare nel range di valori corretto. Bisogna posizionare la retta N/R correttamente rispetto all'OFFSET e all'inclinazione perché altrimenti non potremmo sapere se alle nostre resistenze equivale esattamente quel numero esadecimale. Per ottenere questo si fissano i 2 estremi 8000h e FFFFh ai corrispettivi valori di resistenza. Lo strumento di calibrazione genererà 2 numeri che rappresentano la calibrazione voluta, per quel particolare strumento. Intuitivamente si capisce che saranno sicuramente numeri che vengono sommati e moltiplicati alla retta per farla traslare e scalare. Questi 2 numeri vengono memorizzati alla prima pagina della memoria dati Flash/EE. In laboratorio, al momento della calibrazione, tramite un programma i progettisti hardware si sono occupati della memorizzazione dei 2 numeri.
I primi 2 byte contengono il risultato della calibrazione all'estremo inferiore del nostro range di temperature, cioè 0°C, mentre gli altri 2 byte contengono il risultato all'estremo superiore, cioè 660°C. Ogni volta che lo strumento viene acceso questi valori vengono caricati nei registri dell'AduC816 che riguardano il coefficiente di calibrazione dell'ADC primario, cioè quello che stiamo utilizzando noi. I registri sono: OF0M/H, per il coefficiente di calibrazione relativo a 0°C e GN0M/H per il coefficiente di calibrazione relativo al fondo scala. Si chiamano in questo modo perché OF sta per OFFSET e GN sta per GAIN.
La calibrazione dell'ADuC816 è già stata fatta dalla casa costruttrice ma noi non possiamo utilizzare quella perché le schede e i componenti aggiunti falsano questa calibrazione. La calibrazione tiene conto, infatti, di tutte le componenti dello strumento e in base a questa opera l'aggiustamento.
Nella funzione che inizializza il convertitore ci sarà anche la parte relativa alla calibrazione. Dobbiamo fare una lettura, alla prima pagina e salvare i dati nei registri come abbiamo appena visto.

4.3-Test dello strumento

Per programmare il microcontrollore togliamo l'involucro di plastica e colleghiamo, tramite l'interfaccia seriale, le schede con il computer. A questo punto si esegue il programma WSD caricando il codice nella memoria programmi Flash/EE.
Si possono effettuare diverse prove, in modo graduale per evidenziare, prima i bug più piccoli, fino ad arrivare a testare tutte le potenzialità dello strumento.
Una prima prova può essere quella di non considerare le Pt100 e impostare da software quello che si vuole visualizzare. Facciamo visualizzare, per esempio, i valori estremi del nostro range di temperature e alcuni valori intermedi simulando la lettura da parte dell'ADC.
In questa fase abbiamo trovato un problema che ci ha impegnato molto, prima di poterlo risolvere. Il problema riguardava una caratteristica dell'ADuC816 sulla calibrazione. Prima di tutto, infatti, vanno caricati i coefficienti della calibrazione e poi si setta ADCMODE per far iniziare le conversioni dell'ADC. Le due istruzioni non possono essere invertite nel programma altrimenti la visualizzazione sul display appare errata.
Da queste prima prove emergono anche dei fatti marginali. Il terzo digit, per esempio, deve avere sempre il punto acceso, questo si risolve facendo una maschera con l'OR logico tra il numero che si voleva visualizzare e il numero binario 1000 0000 = 128, perché il punto sta sull'ultimo bit del numero a 8 bit che fa accendere i led del digit.
Una ottimizzazione è quella di non visualizzare gli zeri a sinistra del numero perché esteticamente è più bello. Per fare questo basta inserire degli if-then-else per controllare le cifre che stiamo scrivendo sullo schermo.
A questo punto possiamo attaccare le sonde ed eliminare quella parte di codice che simulava l'acquisizione. Ricarichiamo il programma con WSD e analizziamo quello che viene visualizzato. Dopo aver caricato il programma, la connessione con il computer può essere eliminata, perché il programma è in grado di girare da solo, oppure si può far girare il programma da computer. Quando lo strumento viene acceso, automaticamente, va prima a vedere se ci sono segnali che arrivano dall'esterno, se questi non ci sono allora inizia l'esecuzione del programma, altrimenti tutto viene gestito dall'esterno.
Appena lo strumento inizia la conversione quello che si deve osservare sullo schermo, se non si toccano le sonde, è la temperatura ambiente. Utilizzando una qualsiasi fonte di calore, anche le mani o un'accendino, si vede che la temperatura sale. In alcuni casi, anche se le sonde sono alla stessa temperatura, le temperature visualizzate non sono identiche ma questo è dovuto alla precisione delle sonde.
L'ultima prova riguarda il test dei 3 tasti. L'ultimo bug trovato faceva parte del codice che gestisce il tasto di visualizzazione della temperatura minima. Prima di resettare, almeno una volta, la temperatura minima era sempre zero per una questione di inizializzazioni. Non potendo eliminare le inizializzazioni ho ritardato il programma di pochi millisecondi per aspettare la prima conversione dell'ADC senza però avere effetti indesiderati sul display, come per esempio dover aspettare troppo tempo per la prima visualizzazione di temperatura.
Provando lo strumento, ho visto che quando la sonda era scollegata sul display veniva visualizzato lo zero. Questa decisione, però, era poco corretta perché zero è uno dei valori di temperatura facenti parte del range da noi scelto quindi non può essere utilizzato per una situazione di non misura. Abbiamo fatto delle modifiche software per far visualizzare, al posto di zero, tutti trattini.
Un'altra piccola imprecisione riguardava il fatto di non testare, in nessun momento, il corretto funzionamento di tutti i led dei digit. Abbiamo aggiunto una modifica che invece di visualizzare zero, prima della conversione di partenza, visualizza il risultato dato da tutti i led accesi.

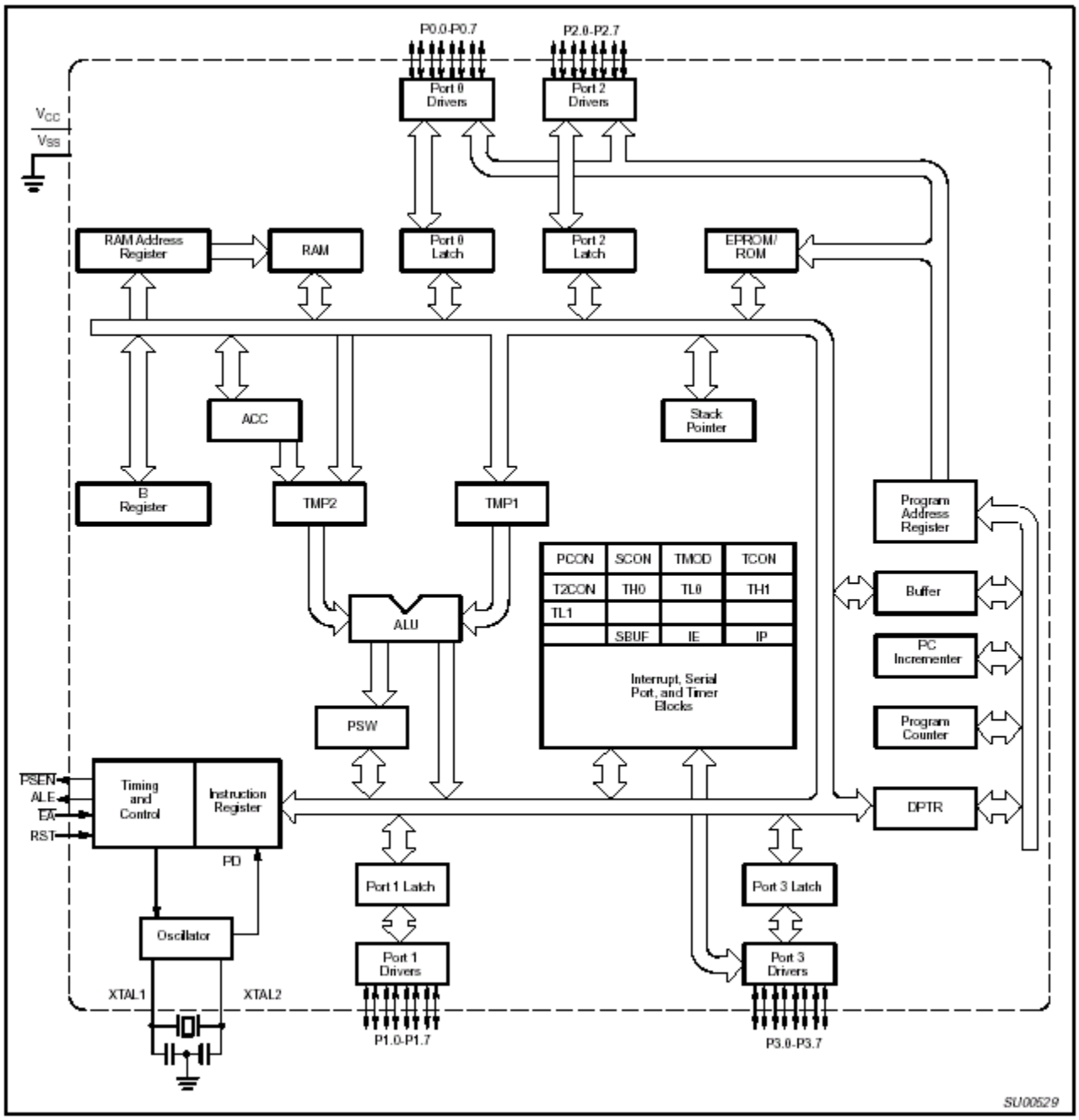
5-Manuale d’uso

Il 2µS1T2 è uno strumento molto facile da utilizzare ed è, come si sente dire spesso, user-friendly.
La foto della facciata frontale dello strumento è l’ultima di paragrafo 4.1.
La facciata dello strumento è composta dal display, da 3 tasti e da 2 led per ogni µS1T2. Sappiamo che l’involucro contiene, per ragioni industriali 2 µS1T2, A e B, quindi tutti questi elementi sono raddoppiati. I 2 led saranno sempre spenti in questa prima versione dello strumento perché sono riservati per usi futuri come allarmi. Sul display, formato da 4 digit, viene visualizzata la temperatura al decimo di grado, temperatura che può andare da 0.0°C a 660.0°C. I pulsanti hanno ognuno uno scopo ben specifico.
La F sta per function e serve per resettare lo strumento dando quindi l’origine dell’asse dei tempi. In futuro si può pensare di utilizzare questo tasto per inserire altre funzionalità allo strumento.
La freccia in su (↑) serve per visualizzare, al posto della temperatura che lo strumento sta misurando al momento, la temperatura più alta dal reset.
La freccia in giù (↓) serve per visualizzare, al posto della temperatura corrente, la temperatura più bassa dal momento del reset.
Dietro all’involucro, che contiene il µS1T2, ci sono dei connettori ai quali si possono collegare 2 Pt100. Questo si deve fare seguendo attentamente l’indicazione sulla targhetta stampata.
Nel caso in cui la Pt100 è scollegata, sul display vengono visualizzati quattro trattini per evidenziare lo stato di non misura. Questo avviene se uno solo o entrambi i connettori della Pt100 sono scollegati.

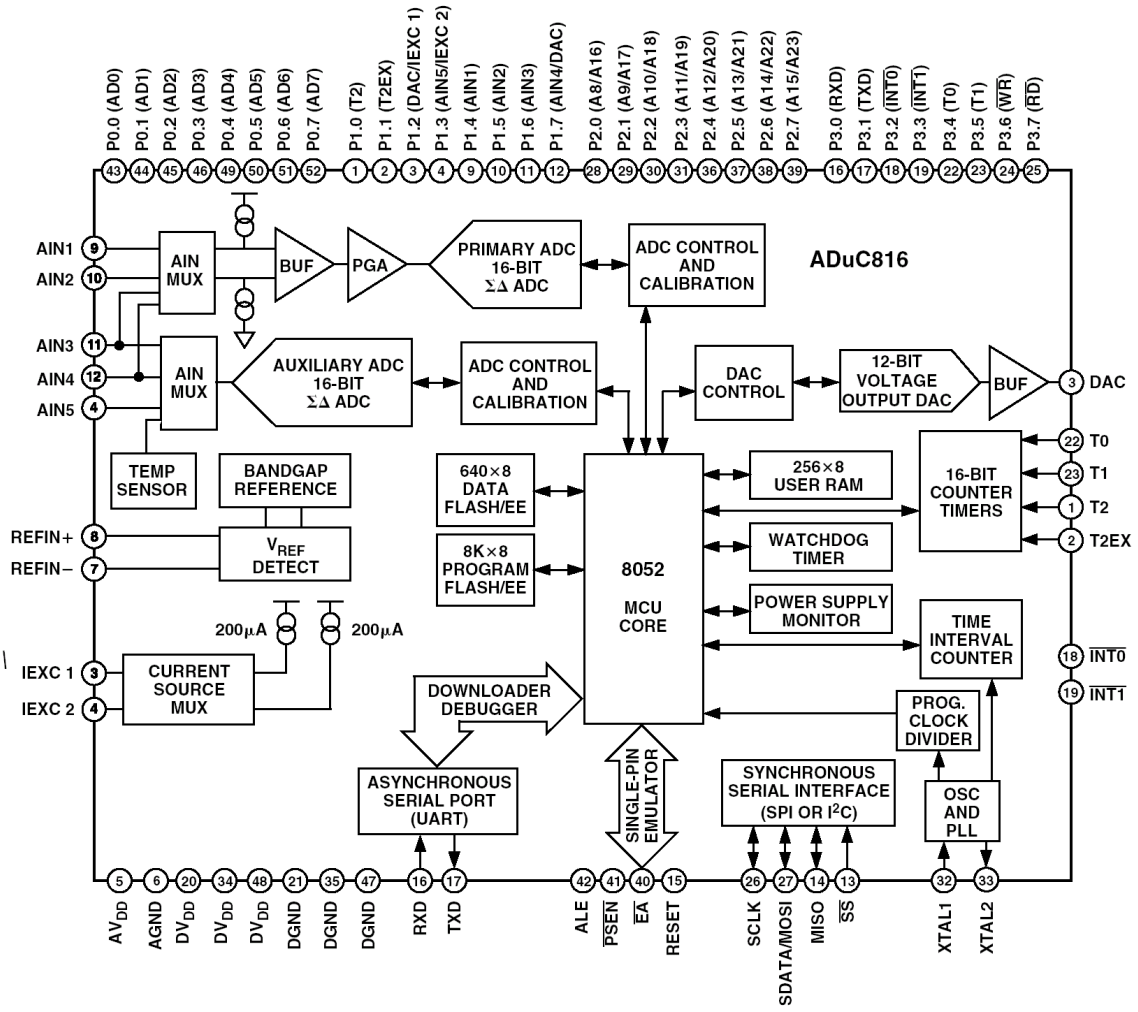
6-Allegati

6.1-Data sheet

6.1.1-Schema dettagliato dell'architettura interna dell'8051



6.1.2-Architettura e piedinatura dell'ADuC816



Notare che il processore è l'8052, derivato dell'8051.

6.2-Codice prototipo base

```
/*prototipo per il calcolo della temperatura a partire da Nx tramite
   la formula di interpolazione.*/
  #include<stdio.h>
  #define X 512
  const float tabella[]={     0.0000,   9.3196,  18.6651,   28.0365, 37.4342,
                             46.8584,  56.3093,  65.7871,  75.2920,  84.8243,
                             94.3842,103.9721,113.5880,123.2322,132.9051,
                            142.6069,152.3378,162.0980,171.8880,181.7078,
                            191.5579,201.4384,211.3498,221.2922,231.2659,
                            241.2713,251.3087,261.3783,271.4805,281.6156,
                            291.7839,301.9858,312.2216,322.4915,332.7961,
                            343.1356,353.5103,363.9207,374.3671,384.8499,
                            395.3695,405.9262,416.5205,427.1527,437.8233,
                            448.5327,459.2813,470.0696,480.8980,491.7669,
                            502.6768,513.6282,524.6215,535.6572,546.7359,
                            557.8580,569.0241,580.2346,591.4902,602.7913,
                            614.1385,625.5324,636.9735,648.4625,660.0000,
                            671.5865};
   /*-------------------------------------prototipi-------------------------------------*/
  int N_precedente(int);
  float T_relativo(int);
  float calcola_Tx(int);
  /*-----------------------------------------------------------------------------------*/
  void main(void)
  {
     int Nx;
     /*3 variabili per la temperatura: due che riguardano la storia Tmax e Tmin
       (temperatura massima e quella minima dal momento del reset) e  Tx che è la
       temperatura della lettura attuale*/
      float Tx,Tmax=-1,Tmin=1000;
     /*3 variabili boleane che indicano quale dei 3 pulsanti è stato
     selezionato e quindi cosa visualizzare: M=massimo, m=minimo, r=reset*/
     int M=0,m=0,r=0;
     int i=0;
     while(i<5)
     {
      printf("Inserire Nx per ottenere Tx (inserire il numero in esadecimale con o senza 0x)");
      scanf("%x", &Nx);
      printf("Inserire il valore del pulsante M(1 o 0)");
      scanf("%d", &M);
      printf("Inserire il valore del pulsante m(1 o 0)");
      scanf("%d", &m);
      printf("Inserire il valore del pulsante r(1 o 0)");
      scanf("%d", &r);
      Tx=calcola_Tx(Nx);
      if(Tx>Tmax) Tmax=Tx;
      if(Tx<Tmin) Tmin=Tx;
```

```
    if(M) printf("Tmax= %f",Tmax);
      else if(m) printf("Tmin= %f",Tmin);
            else printf("Tx= %f",Tx);
     if(r) {Tmax=-1;Tmin=1000;}
     i++;}
}

//ritorna l'N precedente a quello passato in input:Nx
int N_precedente(int Nx)
{   int N=0x8000;
    while(N<=Nx)
       {N=N+X;}
    N=N-X;
    return N;}

//ritorna il valore di temperatura relativo all'N passato in input
float T_relativo(int N)
{   N=N-0x8000;
    return(tabella[N/X]);}

/*calcola il Tx dato Nx tramite la formula di interpolazione*/
float calcola_Tx(int Nx)
{   int Np=N_precedente(Nx);
    float Tp=T_relativo(Np);
    float Ts=T_relativo(Np+X);
    float Tx=Tp+((Ts-Tp)/X)*(Nx-Np);
    return Tx;}
```

6.3-Codice programma

6.3.1-Codice programma principale

```
#include "ADuC816.h"
#define X 512
const float tabella[]={   0.0000,   9.3196,  18.6651,   28.0365,  37.4342,
                          46.8584,  56.3093,  65.7871,  75.2920,  84.8243,
                          94.3842,103.9721,113.5880,123.2322,132.9051,
                         142.6069,152.3378,162.0980,171.8880,181.7078,
                         191.5579,201.4384,211.3498,221.2922,231.2659,
                         241.2713,251.3087,261.3783,271.4805,281.6156,
                         291.7839,301.9858,312.2216,322.4915,332.7961,
                         343.1356,353.5103,363.9207,374.3671,384.8499,
                         395.3695,405.9262,416.5205,427.1527,437.8233,
                         448.5327,459.2813,470.0696,480.8980,491.7669,
                         502.6768,513.6282,524.6215,535.6572,546.7359,
                         557.8580,569.0241,580.2346,591.4902,602.7913,
                         614.1385,625.5324,636.9735,648.4625,660.0000,
                         671.5865};
unsigned char errore=0;
extern unsigned char array[4];
bit M,m,r;
int Nx=0x8000;
/*-------------------------prototipi-----------------------------------------*/
int N_precedente(int);
float T_relativo(int);
float calcola_Tx(int);
void imposta_array(float a);
extern void inizializza_timer0 (void);
void leggi_Nx(void);
void inizializza_convertitore(void);
/*-------------------------------------------------------------------------------*/
void main(void)
{      /*3 variabili per la temperatura: due che riguardano la storia Tmax e Tmin
         (temperatura massima e quella minima dal momento del reset) e  Tx che è la
         temperatura della lettura attuale*/
    float Tx,Tmax=-1,Tmin=1000;
       /*3 variabili boleane che indicano quale dei 3 pulsanti è stato
         selezionato e quindi cosa visualizzare: M=massimo, m=minimo, r=reset*/
    int i=0;
    unsigned int Nx_unsigned;
    M=0; m=0; r=0;
    P0.1=0;
    inizializza_convertitore();
    array[0]=0xFF;
    array[1]=0xFF;
    array[2]=0xFF;
```

```
   array[3]=0xFF;
   inizializza_timer0();
   while(!RDY0);          //ciclo che aspetta la prima conversione
   while(1)
   { leggi_Nx();
     Nx_unsigned=(unsigned int)Nx;
     errore=0;
     if ((Nx_unsigned<0x8000)||(Nx_unsigned>0xFDFF))
       errore=1;
     Tx=calcola_Tx(Nx);
     if(Tx>Tmax)
       Tmax=Tx;
     if(Tx<Tmin)
       Tmin=Tx;
     if(M)
      imposta_array(Tmax);
      else if(m)
            imposta_array(Tmin);
            else imposta_array(Tx);
     if(r) {Tmax=-1;Tmin=1000;}}
}

void inizializza_convertitore(void)
{   ADCMODE=0x00;       // Disabilita ADC
   SF=255;
   ICON=0x03;
   ADC0CON=0x43;
   /*-----------------------------calibrazione---------------------------------*/
   EADRL=0;            //indirizzo di pagina della Flash/EE
   ECON=1;            //comando di lettura dei 4 byte della pagina
   GN0H=EDATA1;        //coefficienti di calibrazione per lo strumento
   GN0M=EDATA2;        //Full-scale rappresenta i 660°C
   OF0H=EDATA3;
   OF0M=EDATA4;        //offset rappresenta gli 0°C
   ADCMODE=0x22;
}

void leggi_Nx(void)
{   if(RDY0)
     {Nx=ADC0H;
      Nx*=256;
      Nx+=ADC0M;
      ADCMODE=0x22;}
}

//ritorna l'N precedente a quello passato Nx
int N_precedente(int Nx)
{   int N=0x8000;
```

```
   while(N<=Nx)
      {N=N+X;}
   N=N-X;
   return N;}

//ritorna il valore di temperatura relativo all'N passato in input
float T_relativo(int N)
{   N=N-0x8000;
   return(tabella[N/X]);}

//calcola il valore di temperatura Tx per quel valore di Nx
float calcola_Tx(int Nx)
{   int Np=N_precedente(Nx);
   float Tp=T_relativo(Np);
   float Ts=T_relativo(Np+X);
   float Tx=Tp+((Ts-Tp)/X)*(Nx-Np);
   return Tx;}

 //assegna ad un array di 4 elementi i valori de visualizzare sui digit: come print
void imposta_array(float a)
{   const unsigned char tabella[] = {  0x3F,   //0
                                       0x06,   //1
                                       0x5B,   //2
                                       0x4F,   //3
                                       0x66,   //4
                                       0x6D,   //5
                                       0x7D,   //6
                                       0x07,   //7
                                       0x7F,   //8
                                       0x67    //9
                                    };
   int e,b=a*10;
   unsigned char c,d;
   if(errore==1)       /*visualiziamo tutti trattini*/
     {array[0]=0x40;array[1]=0x40;array[2]=0x40;array[3]=0x40;}
     else{ array[0]=tabella[b%10];    //modulo:divisione che restituisce il resto
          b/=10;                      //divisione intera
          c=tabella[b%10];
          array[1]=(c | 0x80);         //per accendere il punto della terza cifra
          b/=10;
          d=tabella[b%10];
          b/=10;
          e=(b%10);
          if(e)
            {array[2]=d;
             array[3]=tabella[e];}
            else {array[3]=0x00;
                 if(d==0x3F)
                   array[2]=0x00;
                   else array[2]=d;}}
}
```

6.3.2-Codice programma di visualizzazione

```
#include "ADuC816.h"
unsigned char array[4];
int cont=0;
extern bit M,m,r;
interrupt [0x0B] using[2] void T0_int (void)
{   P2=0;
   switch(cont)
     {case 0:
          P3.7=1;
          P3.4=0;
          break;
      case 1:
          r=P0.0;
          P3.4=1;
          P3.5=0;
          break;
      case 2:
          M=P0.0;
          P3.5=1;
          P3.6=0;
          break;
       case 3:
          m=P0.0;
          P3.6=1;
          P3.7=0;};
   P2=array[cont];
   cont=(cont+1) % 4;
}

void inizializza_timer0 (void)
   #define RELOAD0L   0x00
   #define RELOAD0H   0x00
{
   TMOD=0x12;          //Timer 0 a 8-bit con auto-Reload;
   TR0=1;              //Parte timer 0
   ET0=1;              //Abilita int da timer 0
   PT0=1;              //Interrupt timer 0 priorità 1
   TL0=RELOAD0L;       //Valore di partenza timer 0
   TH0=RELOAD0H;       //Valore di reload timer 0
   EA=1;               //Abilita tutte le interruzioni
}
```