Università degli Studi di Padova

Facoltà di Ingegneria

Corso di Laurea Triennale in Ingeneria dell'Informazione

tesi di laurea

Realizzazione sistema di monitoraggio temperature basato su scheda di sviluppo "Arduino Ethernet"

Relatore: Paolo Tenti
Correlatore: Marco Stellini

Laureando: Stefano Montagner

24 luglio 2012

Indice

Capitolo 1

Introduzione

Nell'elaborato si descrive la realizzazione e la messa in opera di un sistema per il monitoraggio di temperature. Scopo del progetto è l'osservazione della temperatura di otto cavi elettrici situati nel sottotetto dell'edificio DEI/A, utilizzati nell'ambito del progetto riguardante le smart grid. In questo progetto infatti, una rete di informazione affianca la rete di distribuzione elettrica e ne gestisce gli scambi energetici in maniera intelligente. In particolare, per la simulazione della rete di distribuzione pubblica sono stati posati dei cavi che percorrono complessivamente 600mt e corrono lungo il sottotetto dipartimentale la cui pavimentazione è composta da uno strato di lana di vetro. Per motivi di sicurezza risulta necessario monitorare la temperatura esterna dei cavi che, durante il funzionamento, potrebbero riscaldarsi con conseguenze anche gravi.
Al fine di avere massima flessibilità nella consultazione, si è deciso di utilizzare un sistema che possa essere rapidamente e facilmente accessibile attraverso un comune browser web, il che garantisce di monitorare le temperature istantaneamente e attraverso un comune pc. Oltre a questo, si è ritenuto utile mantenere uno storico dei dati, utilizzando MRTG (Multi Router Traffic Grapher). Si tratta di un software che permette la visualizzazione in forma grafica dei dati acquisiti.
La scelta dei sensori ha portato ad un prodotto della Smartec, precisamente al modello SMT160-30, che copre un range di temperature tra -45°C e +130°C. La Smartec produce anche un sistema di acquisizione (SMTA08), a cui è possibile collegare fino ad otto sensori e che comunica con pc o altri dispositivi attraverso una porta seriale.
Infine si è deciso di utilizzare l'Arduino ethernet, una scheda di sviluppo e prototipazione open source basata su microcontrollore, per la presenza di una porta ethernet e per la semplicità di programmazione.
In concreto, le specifiche di progetto sono la creazione di una pagina web

pubblica ed una privata. Nella prima verranno visualizzate le temperature di tutti i sensori, sia in termini numerici, sia con l'ausilio di una grafica attraverso delle barre colorate per indicare il livello di allerta di ogni singolo sensore. Nella pagina privata (cui si può accedere solo mediante autenticazione), è possibile modificare l'indirizzo IP dell'Arduino ethernet, la password della pagina privata e le descrizioni dei sensori, nonché è stata aggiunta la possibilità di variare i quattro valori di soglia che regolano le barre della pagina pubblica.

Capitolo 2

Scelte progettuali

In questo capitolo vengono esposte le scelte effettuate per portare a compimento il progetto. Il primo problema affrontato è l'interfacciamento tra modulo di acquisizione e Arduino. Per far questo, si è deciso di collegare direttamente la seriale di Arduino con i pin del microcontrollore della scheda di acquisizione, in maniera tale da non utilizzare ulteriori adattatori di livelli logici. Successivamente verrà realizzato del codice che, utilizzando la libreria per la comunicazione seriale, legge i dati e li visualizza attraverso delle pagine web, tramite la creazione di un web server. Utilizzando una porta ethernet questo dispositivo può essere installato in qualsiasi rete in maniera veloce. L'ausilio del web server consente di visualizzare i dati tramite un comune browser web, senza quindi dover installare altro software nei client. Infatti, attualmente qualsiasi computer o dispositivo mobile dispone di un browser web.
Un secondo punto affrontato è relativo alla necessità di salvare alcuni dati anche in assenza di alimentazione. Questi sono: i quattro valori di soglia, l'indirizzo IP, la password della pagina di configurazione e le descrizioni

Indirizzo	**Parametro**
0	controllo
10-17	password
20-31	IP
40-51	soglie
150-189	seconda descrizione
...	
450-489	ottava descrizione

Tabella 2.1: Allocazione della memoria EEPROM

dei sensori. A questo scopo verrà utilizzata la memoria EEPROM da 1024 byte di cui è dotato Arduino. Inoltre, è stato inserito un piccolo controllo, in quanto se il primo byte della memoria EEPROM contiene il carattere 'c' allora i dati sono scritti correttamente, al contrario se non è presente vengono inseriti i valori di default. Per semplicità di programmazione

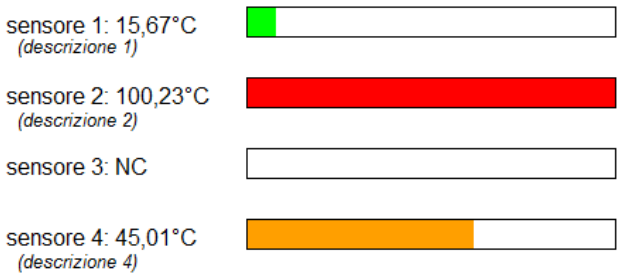


Figura 2.1: risultato finale di quattro sensori

i parametri di configurazione hanno delle restrizioni in quanto le soglie possono essere valori interi positivi di al massimo tre cifre, come l'IP. Per quanto riguarda la password la sua lunghezza è fissa a otto caratteri, mentre le descrizioni possono avere lunghezza variabile fino ad un massimo di quaranta caratteri. Infine, vengono riportati in tabella (2.1) le collocazioni dei dati in memoria.
Per quanto riguarda l'unita di misura della temperatura è stato adottato il grado Celsius in quanto è quella predefinita sui sensori e di più largo uso comune, anche se volendo era possibile adottare altre unità di misura utilizzando le formule di conversione.
Nella pagina pubblica, come detto nell'introduzione, dovranno essere presenti i valori numerici delle temperature oltre a delle barre colorate per indicare il livello d'allerta in modo più immediato. E' stato, quindi, deciso di regolare l'avanzamento della barra utilizzando quattro valori di soglia:

- VERDE: temperatura sotto controllo;
- GIALLO: temperatura appena oltre il valore normale;
- ARANCIONE: temperatura critica;
- ROSSO: temperatura fuori controllo.

Nella pagina privata vi si può accede solo mediante autenticazione, a questo scopo è stato implementato un sistema di autenticazione denominato base e supportata da tutti i browser. Nella pagina privata è possibile modificare

tutti i parametri di configurazione grazie a delle semplici caselle di testo dove inserire i valori desiderati.
Particolare attenzione è stata posta al caso in cui venissero accidentalmente dimenticati l'indirizzo IP o la password impostati, in quanto non sarebbe più possibile accedere alle pagine web. Per questo motivo è stato deciso di collegare un pulsante di reset, in modo tale da cancellare la memoria e riportare i valori predefiniti di tutti i parametri. In questo modo però, sarebbe possibile premere involontariamente il pulsante. Per evitare il problema si è creato un sistema per cui bisogna collegare l'alimentazione di Arduino con il pulsante reset premuto, evitando così reset accidentali.

Capitolo 3

Descrizione hardware

La parte hardware interessa il rilevamento, elaborazione e la comunicazione ethernet. Nello sviluppare questa applicazione si è cercato di ridurre al minimo la progettazione di schede, componenti o quant'altro. Per questo è stata utilizzata una scheda di acquisizione realizzata dalla stessa casa produttrice dei sensori (Smartec) e per quanto riguarda l'elaborazione dei dati e la visualizzazione web è stato impiegato un Arduino ethernet.
In figura 3.1 viene riportato uno schema funzionale dell'apparecchiatu-

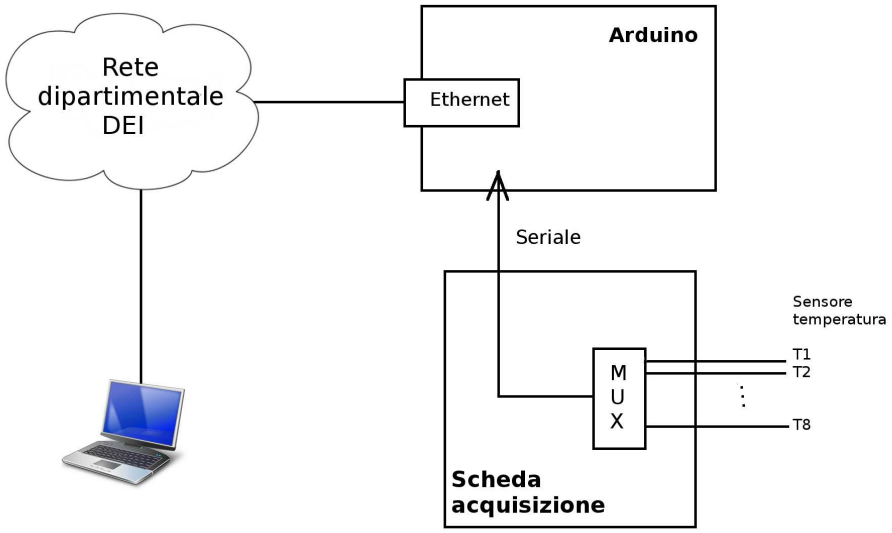


Figura 3.1: schema funzionale complessivo

ra realizzata. Utilizzando un multiplexer il sistema di acquisizione legge le temperature e le invia alla scheda Arduino attraverso una connessione seriale, a sua volta la scheda Arduino è connessa alla rete ethernet dipartimentale. Tramite un pc connesso alla rete si potrà così accedere alle pagine web contenenti le temperature.

3.1 Arduino Ethernet

Arduino è una scheda di sviluppo e prototipazione basata su microcontrollore. Quest'ultimo è un dispositivo elettronico integrato su singolo chip, nato come evoluzione alternativa al microprocessore ed utilizzato generalmente in sistemi embedded, ovvero per applicazioni specifiche di controllo digitale. Arduino si basa su hardware e software flessibili e facili da usare per artisti, designer e hobbisti ed è progettato per rendere il processo di utilizzo dell'elettronica in progetti multidisciplinari più accessibile. Il nome Arduino deriva da quello di un bar di Ivrea (che richiama a sua volta il nome di Arduino d'Ivrea, primo re d'Italia dal 1002 al 1014) frequentato da alcuni dei fondatori del progetto.
Arduino ormai non è più solo una scheda ma un progetto che comprende svariate schede con moltissime caratteristiche in comune, basate su microcontrollore con PIN connessi alle porte I/O, un regolatore di tensione e generalmente un'interfaccia usb per la comunicazione con il computer e per la programmazione.
Tutto l'hardware prodotto dal team di Arduino è open source, distribuito nei termini della licenza Creative Commons Attribution-ShareAlike 2.5, quindi tutte le informazioni sull'hardware e i progetti relativi sono reperibili in rete in modo che ognuno possa creare il proprio clone o una versione modificata in base alle esigenze. Grazie alle librerie software disponibili è stato possibile connettere ad Arduino ogni tipo di oggetto elettronico, tra cui computer, sensori, attuatori e display. Questa possibilità ha consentito lo sviluppo di prodotti compatibili da parte di piccole e medie aziende in tutto il mondo, e infatti oggi è possibile scegliere tra un'enorme quantità di schede Arduino compatibili. Il progetto è completamente italiano in quanto venne avviato ad Ivrea nel 2005 e tuttora le schede originali vengono prodotte in Italia dalla Smart Projects, i cloni invece sono prodotti in tutto il mondo.
Benché i progetti hardware e software sono resi disponibili con licenze open source, il team di sviluppo ha espresso il desiderio che il nome Arduino venga riferito solo al prodotto originale.
Arduino ethernet è basato sul microcontrollore ATmega328 e possiede 14

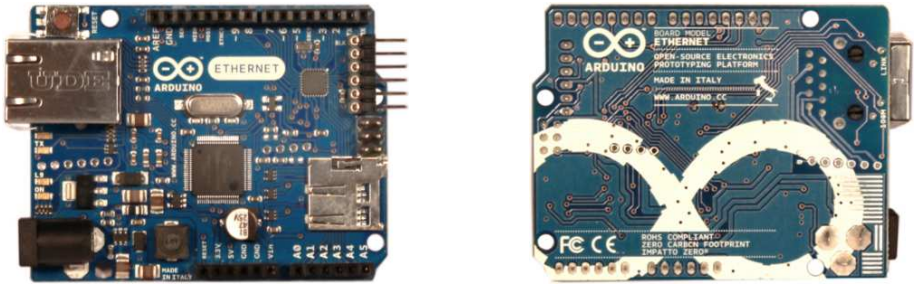


Figura 3.2: scheda Arduino ethernet fronte e retro

pin digitali di input/output, 6 input analogici, un oscillatore a cristallo da 16MHz, un connettore di alimentazione,connettore RJ45 e un lettore di memorie microSD.
Il microcontrollore ATmega328 dispone di 32KB di memoria flash per la programmazione, 2KB di SRAM e una EEPROM da 1KB. La scheda può essere alimentata attraverso Power over Ethernet, una tecnica che permette di alimentare apparecchiature utilizzando lo stesso cavo che le collega alla rete dati ethernet, (soluzione parzialmente adottata nel nostro caso) o attraverso un alimentatore esterno con tensione continua compresa tra 6 e 18 volt, consigliati tra 7 e 13.
Per l'applicazione descritta in questa tesi è stato utilizzato un Arduino originale senza alcun tipo di modifica, in particolare l'Arduino ethernet in quanto già provvisto di una porta ethernet (figura 3.2). Questa scheda differisce dalle altre, oltre che per la presenza di un connettore RJ45 e al relativo driver ethernet, anche per la mancanza di una porta usb. Per la programmazione si usa in alternativa l'USB Serial Light Adapter, un adattatore usb/seriale prodotto sempre dal team di Arduino illustrato in figura 3.3.

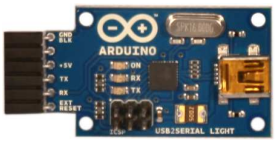


Figura 3.3: USB Serial Light Adapter

Unico aspetto limitativo, ma che non influenza tuttavia il progetto, è che il modulo ethernet montato su Arduino supporta massimo quattro connessioni simultanee.
Tutti i pin disponibili all'esterno sono utilizzabili fatta eccezione per i pin 10,11,12 e 13 che sono riservati all'interfacciamento con il modulo ethernet

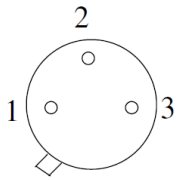
e non possono essere utilizzati per altro. I pin 0(Rx) e 1(Tx) vengono collegati all'adattatore seriale-USB per la trasmissione seriale con il computer e per la programmazione. Tra i pin di I/O 4 hanno la possibilità di essere programmati per avere un'uscita PWM(Pulse-width modulation), un tipo di modulazione digitale che permette di ottenere una tensione media variabile dipendente dal rapporto tra la durata dell' impulso positivo e di quello negativo. Inoltre, è presente un pulsante per far riavviare il microcontrollore, ma tale pulsante non verrà utilizzato.
Per quanto riguarda le dimensioni fisiche la massima lunghezza è di 6.9cm mentre i pin sono collegati sulla parte superiore della scheda attraverso dei connettori femmina da 0.1

3.2 Sensori (SMT160-30)

L'SMT160-30 è un sensore di temperatura integrato a tre terminali con un'uscita a duty cycle variabile. Il duty cycle è, in presenza di un segnale a forma d'onda rettangolare, il rapporto tra la durata del segnale alto e il periodo totale del segnale.

$$d = \frac{t}{T}$$

t porzione di periodo a livello alto
T è il periodo totale
Due terminali del sensore sono utilizzati per l'alimentazione a 5V, mentre il terzo per il segnale d'uscita. Quest'ultimo è una forma d'onda rettan-



bottom view

1 Output
2 + Vcc
3 GND

Figura 3.4: vista del sensore da sotto con descrizione pin

golare in cui il duty cycle dipende linearmente dalla temperatura, secondo

l'equazione:

$$d = 0.320 + 0.00470t$$

d duty cycle
t temperatura in °C

Due semplici esempi:
$t = 0$°C $d = 0.320$ o 32%
$t = 130$°C $d = 0.931$ o 93.1%

La modulazione di duty cycle dell'uscita è usata perché può essere interpretata anche da microcontrollori senza convertitori analogico-digitali. Nelle specifiche, il costruttore fornisce dei parametri che permettono di individuare che: la risoluzione massima del sensore è minore di 0.005°C mentre l' accuratezza assoluta, cioè il massimo scostamento tra la misura fornita dal sensore e il valore vero della temperatura, è di 0.7°C nel range di temperature da -30°C a +100°C e di 1.2°C nel range da -45°C a +130°C. La non linearità è la deviazione dal miglior fit sull'intero range di temperature e per questo sensore corrisponde a 0.2°C.
Generalmente l'uscita del sensore viene collegata direttamente ad un ingresso di un microcontrollore. Quest'ultimo può determinare il duty-cycle attraverso un campionamento dell'ingresso. Se il microcontrollore non è abbastanza veloce può andare a campionare anche un periodo successivo e questo comporta un errore stimato con la formula seguente:

$$T_{err} = \frac{200 t_s}{\sqrt{6 t_m t_p}}$$

T_{err} errore di misurazione
t_s periodo di campionamento
t_m tempo di misurazione totale
t_p periodo del segnale di output dal sensore
Questo errore verrà calcolato per il sistema di acquisizione della Smartec nel capitolo successivo.
Per facilità di utilizzo il sensore SMT160-30 viene incapsulato all'interno di un cavo lungo cinque metri (figura 3.5) all'estremità vengono portati i pin di alimentazione e di uscita del sensore attraverso i cavetti di colore marrone (Vcc), bianco (Output), verde (Gnd).

Figura 3.5: sensore incapsulato

3.3 Scheda di acquisizione (SMTAS08)

SMTAS08 sta per SMart Temperature Acquisition System cioè un sistema di acquisizione di temperature a otto canali per sensori della Smartec modello SMT160. Questi ultimi come detto nel capitolo precedente hanno un'uscita con duty cycle variabile. Il sistema di acquisizione rileva il periodo del segnale e il tempo in cui è a livello logico alto.

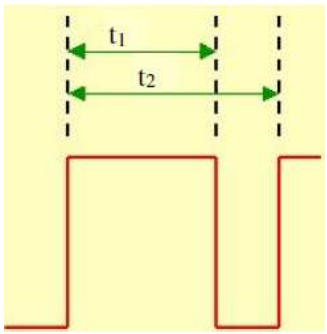


Figura 3.6: segnale di output dal sensore

Dopo aver misurato t_1 e t_2 (figura 3.6) è possibile calcolare la temperatura in gradi Celsius utilizzando la seguente formula:

$$\theta = \frac{t_1}{0.0047 t_2} - 68.09$$

L'SMTAS08 è equipaggiato con un microcontrollore della famiglia Microchip in particolare il PIC16F876 e una porta RS232 per la comunicazione

con computer o altri dispositivi.

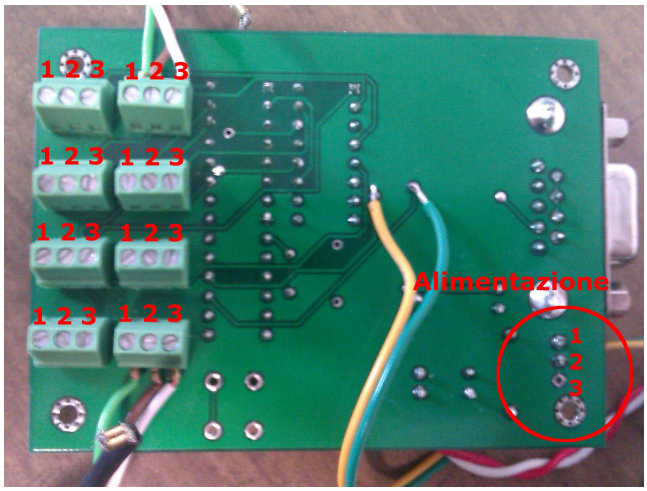
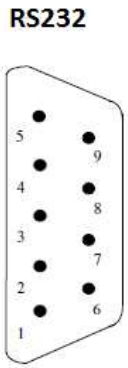


Figura 3.7: scheda di acquisizione vista da sotto

Un multiplexer analogico viene controllato dal microcontrollore per selezionare una delle otto uscite dei sensori e connetterla con la porta di ingresso del microcontrollore in modo tale da poter leggere i valori di t_1 e t_2.
Oltre alla seriale ha altri nove connettori, uno è per l'alimentazione e gli altri uno per ogni sensore. Come si vede dalla figura 3.7 il pin centrale dell'alimentazione è quello positivo mentre i due esterni sono per la massa. Nei connettori dei sensori il pin 1 è la massa, il pin 2 l'alimentazione del sensore e il pin 3 l'output.
Per quanto riguarda i pin della seriale RS232 (figura 3.8) si può vedere



1	NC	Not connected
2	TI	Serial data from PC to board
3	RI	Serial data from board to PC
4	NC	Not connected
5	GND	Ground
6	NC	Not connected
7	NC	Not connected
8	NC	Not connected
9	NC	Not connected

Figura 3.8: descrizione pin dei vari connettori

come sono mappati. L'alimentazione accettata da questa scheda è compresa tra 7 e 18 Volt con una corrente minima di 20mA. I parametri della seriale sono 57600bps (bit per secondo), lunghezza dei dati di 8 bit, un bit di stop e nessun tipo di controllo di parità.
Collegando la scheda di acquisizione al pc e utilizzando Hyper terminal su Windows o programmi simili si possono vedere i dati in uscita dalla seriale. Per avere un help dei comandi basta premere "h" e verrà visualizzato un menù come quello di figura 3.9. Premendo "m" o un numero compreso tra "1" e "8" la scheda inizierà la lettura delle temperature.

```
SMTAS8 TEMPERATURE
MEASUREMENT SYSTEM
    Version :

m -> Infinite measurement cycle
1 -> First sensor, single measurement
2 -> First two sensors, single measurement
3 -> First three sensors, single measurement
4 -> First four sensors, single measurement
5 -> First five sensors, single measurement
6 -> First six sensors, single measurement
7 -> First seven sensors, single measurement
8 -> First eight sensors, single measurement
s -> Stop

Please Make a Choice :
```

Figura 3.9: menu di help sulla seriale

Per ogni sensore la scheda invia al pc i valori di t_1 e t_2 espressi con quattro valori esadecimali Ascii separati da uno spazio e seguiti da "/n" o "CRLF" alla fine della sequenza. In caso che il sensore non sia connesso verrà inviato "FFFF" sia per t_1 che per t_2, nella figura 3.10 è mostrato un esempio di output.

```
1B36 3F92 1AEF  3ED2  1A6B 3D95 1AFA 3EC3  1BFB 413A 1B08 3EF9 19FB 3C7C  FFFF FFFF
1B34 3F89 1AF1  3ED3  1A65  3D8F 1AFB 3ECD 1BFA 4138 1B13 3F14  19FA 3C7E  FFFF FFFF
1B34 3F8E 1AED 3ECD 1A68  3D91 1AF6  3EC2  1BF7  4130 1B15 3F15  19F6  3C7A FFFF FFFF
 \_____/   \_____/   \_____/   \_____/   \_____/   \_____/   \_____/   \_____/
Sensor 1  Sensor 2  Sensor 3  Sensor 4  Sensor 5  Sensor 6  Sensor 7  Sensor 8
```

Figura 3.10: esempio di output dalla seriale con sensore 8 disconnesso

Come detto nel paragrafo precedente è possibile calcolare l'errore di misurazione. Considerando una frequenza di campionamento di 5MHz e una durata di circa 10.5ms allora l'errore è di 0.015°C molto inferiore alla accuratezza massima del sensore.
Per quanto riguarda la seriale i pin 17 e 18 del microcontrollore, rispettivamente Tx e Rx vengono collegati ad un integrato (ICL232CBE) che alza i livelli di tensione passando dai 0V - 5V alle tensioni dello standard RS232.

3.4 Comunicazioni e collegamenti

Nei precedenti paragrafi sono state illustrate le caratteristiche tecniche ed il funzionamento del sistema Arduino e della scheda di acquisizione, nel seguito si riportano i dettagli circa la comunicazione tra i due dispositivi. Arduino dispone di una seriale con livelli di tensione a 0V e 5V mentre quella della scheda di acquisizione è una seriale RS232 con i relativi livelli di tensione che possono variare in valore assoluto tra i 5V e i 25V (quest'ultimo valore ridotto a 13 in alcune revisioni dello standard).

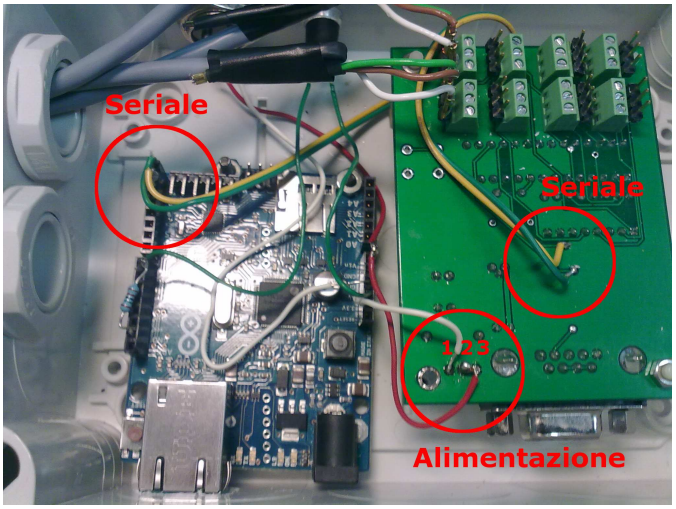


Figura 3.11: Arduino e scheda di acquisizione installate all'interno della scatola

Tuttavia, questi livelli di tensione si impiegano per comunicazioni su tratte piuttosto lunghe e sfruttano dei sistemi a pompa di carica. All'ingresso dell'integrato cui sono collegati il trasmettitore e il ricevitore seriale i livelli logici sono sempre di 0V e 5V. Data la particolare struttura e per evitare ulteriori convertitori, si è deciso di collegare direttamente la seriale del

microcontrollore con quella di Arduino. Per far questo sono stati saldati due cavetti uno verde e uno giallo (vedi figura 3.11) direttamente dalla scheda di acquisizione ai pin 0 e 1 di Arduino. Il filo verde viene collegato al trasmettitore del microcontrollore cioè al pin 17 mentre quello giallo al ricevitore quindi al pin 18. All'altro capo i cavi vengono collegati al pin 0(Rx) dell'Arduino quello verde e al pin 1(Tx) il cavetto giallo.
In questo modo si effettua la comunicazione direttamente tra Arduino e il microcontrollore senza passare per convertitori di livelli di tensione.

3.5 Realizzazione finale

Si riporta nel seguito una descrizione del sistema hardware realizzato.
Come già anticipato è prevista la possibilità di azzerare i parametri di configurazione, a tale scopo è stato predisposto un pulsante collegato tramite due fili di colore verde scuro al pin 8 di Arduino e ai 5V. Una resistenza di pull-down è poi connessa tra massa e il pin 8, in modo tale da mantenere quest'ultimo a massa quando il pulsante è aperto e limitare la corrente sul pin 8 quando è premuto (vedi figure 3.11 e 3.12).
Premendo il pulsante il pin 8 viene portato a livello logico alto, così grazie ad un software appropriato sarà possibile riconoscere la variazione di tensione e azzerare le configurazioni.
Per fornire alimentazione alle schede è stato montato un connettore esterno

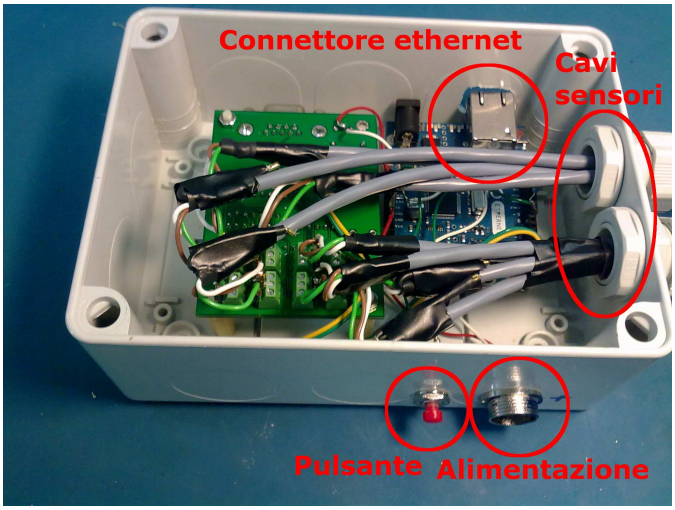


Figura 3.12: Arduino e scheda di acquisizione installate all'interno della scatola

dove sono stati saldati due fili. Uno rosso collega +12V dell'alimentatore alla Vin di Arduino e all'alimentazione della scheda di acquisizione, l'altro, bianco, costituisce la massa dell'alimentatore e dei due dispositivi.
Per finire è stato inserito tutto all'interno di una scatola in plastica (figura 3.12) in modo tale da isolare i componenti dall'esterno e permettendo il collegamento solo del connettore per l'alimentazione e per i sensori.

Capitolo 4

Descrizione software

In questo capitolo verrà descritta la parte software. In particolare l'ambiente di sviluppo di Arduino, utilizzabile per qualsiasi scheda Arduino. Verranno anche descritte le librerie utilizzate nel progetto e loro utilizzo, si darà, poi, una breve descrizione del software MRTG. Per finire verrà esposto e commentato il codice scritto per Arduino, in modo tale da poter comunicare con la scheda di acquisizione e creare le pagine web.

4.1 Ambiente di sviluppo Arduino

L'ambiente di sviluppo integrato (IDE) di Arduino è un'applicazione scritta in Java, derivante da quella creata per il linguaggio di programmazione Processing. L'ide è anche multipiattaforma infatti, può essere utilizzato su Windows, Linus e Mac OS-X. Questo ambiente di sviluppo è stato concepito per utilizzatori a digiuno dallo sviluppo software quindi semplice e intuitivo.
Per semplificare la stesura del codice, l'editor di testo, incluso all'interno dell'IDE, integra il controllo delle parentesi, l'indentazione automatica e il syntax highlighting. Quest'ultimo è la possibilità di visualizzare del testo con differenti colori e font in base alle regole sintattiche del linguaggio di programmazione. Grazie a questo editor è inoltre possibile compilare ed eseguire il software creato, senza l'utilizzo della riga di comando o di programmi esterni. L'ambiente di sviluppo di Arduino include una libreria scritta in C che rende molto più semplice l'implementazione delle operazioni di input/output. Per poter creare un file eseguibile non è richiesta la scrittura di un programma in C ma solo della definizione di due funzioni:

- setup(): funzione chiamata una sola volta all'inizio del programma per definire i settaggi iniziali;

- loop(): funzione invocata ripetutamente, dopo la funzione setup(), che si interrompe solo con lo spegnimento della scheda Arduino.

Per questa applicazione è stata utilizzata la versione 1.0 dell'ambiente di sviluppo.

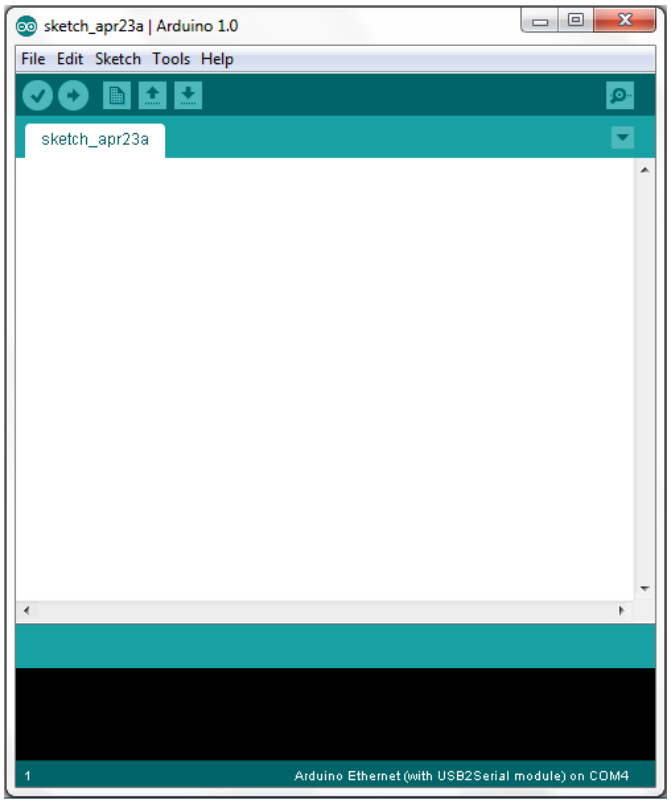


Figura 4.1: ambiente di sviluppo Arduino versione 1.0

Per iniziare verranno analizzati i pulsanti presenti nella barra degli strumenti da sinistra a destra, visibili in figura 4.1:

- Verify: compila il programma;
- Upload: compila e scrive il programma sulla scheda Arduino;
- New: apre un file vuoto;
- Open: apre un file già esistente;
- Save: salva il file corrente;

- Serial Monitor: apre un monitor della porta seriale (viene spesso usato per leggere i dati che l'Arduino invia tramite seriale, se è programmato per farlo).

Qundi si analizzano i menù File, Edit, Sketch, Tools, Help. Il menù File presenta le stesse funzioni dei pulsanti sopra descritti. In particolare, per vedere dei programmi di esempio si può usare il sottomenù "examples", dove sono suddivisi per categoria molti codici. Su "Preferences" si possono settare alcune opzioni dell'ambiente di sviluppo, in generale verrà lasciato tutto invariato. Il menù Edit permette di copiare il codice in formato HTML o in un formato adatto ad essere inserito in un forum. Presenta inoltre la possibilità di inserire dei commenti, a cercare delle parole e di inserire o eliminare l'indentatura nel codice. Tralasciando i primi due pulsanti già noti, il menù Sketch permette di importare una libreria all'intero del programma. Cliccando su "Show Sketch Folder" si potrà aprire la cartella in cui sono salvati tutti i programmi. Nel menù Tools sono presenti una serie di funzioni molto importanti come: "Auto format" sistema la formattazione del codice, "serial monitor" avvia il monitor seriale (stessa funzione già vista in precedenza), "Serial port" contiene un elenco di tutte le periferiche seriali, "Board" è un menù nel quale si seleziona il modello della scheda Arduino utilizzata. Il menù "Burn bootloader" ha il compito di inserire il bootloader nel microcontrollore (ATmega) presente nella scheda Arduino. Se si è stata comprata una scheda Arduino non è necessario inserire alcun bootloader nel microcontrollore poiché e già presente di fabbrica. Spesso, però, può nascere la necessità di acquistare un altro ATmega il quale potrebbe non avere il bootloader precaricato. In quest'ultimo caso è necessario inserirlo manualmente mediante quel menù. Infine il menù Help contiene una serie di risorse utili riguardanti l'Arduino e l'ambiente di sviluppo, ovviamente sono tutte risorse gratuite.

4.2 Librerie utilizzate

4.2.1 PROGMEM

PROGMEM è un modificatore di variabile e impone al compilatore di scrivere alcuni dati in memoria flash, invece che in SRAM, dove andrebbero salvati normalmente. PROGMEM fa parte della libreria pgmspace.h. Quindi, è necessario includere la libreria in questo modo:

```
#include <avr/pgmspace.h>
```

La sintassi di utilizzo è la seguente:

```
TipoDato nomeVariabile[] PROGMEM={dato0,dato1,..}
```

4.2.2 Libreria EEPROM

La EEPROM , acronimo di Electrically Erasable Programmable Read-Only Memory, è un tipo di memoria non volatile, usata nei computer e altri dispositivi elettronici per memorizzare piccole quantità di dati che devono essere mantenuti quando viene tolta l'alimentazione. Le operazioni di scrittura, cancellazione e riscrittura hanno luogo elettricamente. Per leggere un byte si utilizza la funzione "EEPROM.read(indirizzo)", mentre la scrittura avviene mediante la funzione "EEPROM.write(indirizzo,valore)".

4.2.3 Libreria SPI

SPI sta per Serial Peripheral Interface ed è un protocollo seriale sincrono, utilizzato dai microcontrollori per comunicare rapidamente con uno o più dispositivi periferici a breve distanza. Può anche essere utilizzato per la comunicazione tra due microcontrollori. Per questa applicazione la libreria SPI serve a far comunicare il microcontrollore con il driver ethernet, e verrà utilizzato dalla libreria Ethernet quindi non direttamente nel codice.

4.2.4 Libreria Ethernet

Con la scheda Arduino Ethernet, questa libreria permette la connessione a internet. Può essere utilizzata come server, per ricevere connessioni in ingresso o come client, creando connessioni in uscita. La libreria supporta fino a quattro connessioni contemporaneamente (in entrata o in uscita o combinazione).
La libreria ethernet possiede svariate funzioni ma per questo progetto sarà la libreria WebDuino ad utilizzarle. L'unica funzione usata direttamente è "Ethernet.begin(mac,ip)", dove "mac" deve essere un array di 6 byte e "ip" un array di 4 byte.

4.2.5 Libreria Webduino

Webduino è una libreria che permette l'implementazione di un web server, avente la possibilità del passaggio di parametri in GET e POST e l'implementare di un'autenticazione base. Per questo progetto è stata utilizzata la versione 1.7 scaricabile all'indirizzo: https://github.com/sirleech/Webduino
Per dichiarare ed inizializzare il web server sulla porta 80 si utilizza la seguente sintassi:

```
#define PREFIX="";
WebServer webserver(PREFIX,80)
```

Si analizzeranno, ora, le funzioni utilizzate nel progetto:

- `addCommand("nomePagina",&funzione)`

  Crea una pagina web il cui indirizzo è "path/nomePagina" e il cui contenuto è creato dall'output della funzione "funzione" che deve avere la seguente firma: void funzione(WebServer &server, WebServer::ConnectionType type, char *, bool).

- `begin()`

  Avvia il server web.

- `processConnection(buff,&len)`

  Verifica se ci sono client che richiedono una pagina e gli restituisce quest'ultima. La variabile buff è un array di char normalmente di lunghezza 64 e len un intero contenente la lunghezza di buff.

- `print(contenuto)`

  Scrive nella pagina web "contenuto" che può essere un qualsiasi tipo di dato.

- `printP(progStr)`

  Scrive nella pagina web "progStr" che è una stringa contenuta nel PROGMEM dichiarata utilizzando: P(nomeVariabile)=stringa

- `checkCredentials(pass64)`

  Controlla se il nome utente e la password dell'autenticazione base sono corrette. La stringa pass64 è la conversione in base 64 della stringa "nome:password".

- `readPOSTparam(nome,lungNome,valore,lungValore)`

  Legge i parametri in POST dove nome e valore sono due array di char, il primo è il nome della variabile e il secondo il contenuto. Gli altri due parametri sono le rispettive lunghezze. Per verificare se ci sono parametri in POST si controlla che la variabile type ,contenuta nella firma della funzione, è uguale a WebServer::POST.

4.3 Multi Router Traffic Grapher

Multi Router Traffic grapher (MRTG) è un software libero, il cui scopo è il monitoraggio e la misurazione del carico di rete. Inizialmente sviluppato da Tobias Oetiker e Dave Randt per il monitoraggio del traffico dei router, è stato esteso al punto da essere in grado di mostrare grafici e statistiche per quasi tutto. Questo software, scritto in Perl e multipiattaforma, permette la creazione di grafici che visualizzando l'andamento di un determinato fenomeno nel tempo. Per ricavare i dati può utilizzare il protocollo SNMP (Simple Network Management Protocol) o può essere configurato in modo da eseguire uno script o un comando che legga i dati. In questo progetto è stata utilizzata quest'ultima modalità, creando una pagina web su Arduino dove viene resa disponibile la temperatura di un dato sensore, che lo script preleverà.
MRTG, tipicamente, raccoglie i dati ogni cinque minuti, creando poi una pagina HTML per dispositivo, ognuna con quattro grafici. I risultati vengono mostrati con il tempo in ascissa e la risoluzione dei quattro grafici è il giorno, la settimana, il mese e l'anno. In aggiunta, la pagina web mostra anche il massimo, la media e il valore corrente dell'ingresso. MRTG è inoltre in gradi di inviare notifiche via e-mail se i dispositivi presentano valori al di sopra di una soglia prefissata.

4.4 Presentazione codice

In questo paragrafo verrà fatta una descrizione dettagliata del codice scritto per Arduino, in modo tale da poter comunicare con la scheda di acquisizione e per creare il web server.

4.4.1 Librerie e dichiarazione variabili

Per prima cosa verranno importate le librerie viste nei paragrafi precedenti.

```
#include <EEPROM.h>
#include "SPI.h"
#include "Ethernet.h"
#include "WebServer.h"
#include <avr/pgmspace.h>
```

Poi si crea un template, così facendo in futuro per chiamare la funzione print della libreria webServer basterà utilizzare "«", per rendere la scrittura e la lettura del codice più snella.

```
template<class T>
inline Print &operator <<(Print &obj, T arg)
{ obj.print(arg); return obj; }
```

Infine, verranno dichiarate tutte le variabili globali tra cui l'array che contiene il MAC della scheda Arduino, l'array delle temperature rilevate dai sensori, i parametri che vengono caricati all'avvio e un array di caratteri, residente in memoria flash utilizzato dalla funzione che esegue la codifica in base 64.

```
static uint8_t mac[]={0x90,0xA2,0xDA,0x00,0x66,0xBA};
#define PREFIX ""
WebServer webserver(PREFIX, 80);
double temperature[8];
int esp[4]={1,16,256,4096};
int molt[3]={100,10,1};
int tempoRilettura=0;
int soglie[4];
char* pass="admin:password";
char* pass64="YWRtaW46cGFzc3dvcmQ=";
char descrizioni[8][40];
const char base64[] PROGMEM="ABCDEFGHIJKLMNOPQRSTUVW
                "XYZabcdefghijklmnopqrstuvwxyz"
                                    "0123456789+/";
```

4.4.2 Funzione encode

La funzione encode è tratta da una libreria che permette la codifica e decodifica da intero a base 64, in quanto importare tutta la libreria avrebbe richiesto troppo spazio nella memoria di programmazione di Arduino. Questa libreria permette la codifica da base 10 a base 64 di un numero intero. I parametri della funzione sono, in ordine, un array di caratteri contenente la rappresentazione in base 10 del numero da convertire e la rispettiva lunghezza. Gli ultimi due parametri, invece, sono la stringa convertita in base 64 e la rispettiva lunghezza.

```
int encode(char *src,unsigned s_len,
                    char *dst,unsigned d_len)
{
  unsigned triad;
  for(triad = 0; triad < s_len; triad += 3) {
  unsigned long int sr;
```

```
  unsigned i_byte;
  for(i_byte=0;(i_byte<3)&&(triad+i_byte<s_len);
            ++i_byte)
  {
    sr <<= 8;
    sr |= (*(src+triad+i_byte) & 0xff);
  }
  sr <<= (6-((8*i_byte)%6))%6;
  if(d_len < 4) return 1;
  *(dst+0) = *(dst+1) = *(dst+2) = *(dst+3) = '=';
  switch(i_byte) {
    case 3:
      *(dst+3)=pgm_read_byte_near(base64+(sr&0x3f));
      sr >>= 6;
    case 2:
      *(dst+2)=pgm_read_byte_near(base64+(sr&0x3f));
      sr >>= 6;
    case 1:
      *(dst+1)=pgm_read_byte_near(base64+(sr&0x3f));
      sr >>= 6;
      *(dst+0)=pgm_read_byte_near(base64+(sr&0x3f));
    }
    dst += 4;
    d_len -= 4;
  }
  return 0;
}
```

4.4.3 Funzioni per lettura e conversione dati dalla seriale

Per leggere i dati provenienti dalla seriale viene invocata la funzione leggi(), questa invia al modulo di acquisizione il valore "8", così facendo riceverà i valori per tutti gli otto sensori. All'interno del ciclo esterno, compiuto otto volte, viene chiamata quattro volte la funzione leggiDato(), che attende fino a quando è disponibile un dato sulla seriale, successivamente lo legge e lo restituisce. Cosi facendo sarà stata letta una sequenza di quattro byte relativi a t_1 del primo sensore. Il primo dato è seguito da uno spazio, quest'ultimo quindi verrà letto e salvato in una variabile cestino. Vengono quindi ripetuti questi passaggi per leggere t_2 e, richiamando la funzione

converti(String s), verranno convertiti entrambi da esadecimale ad intero. Come ultima operazione, utilizzando la formula presente sul datasheet del sistema di acquisizione, si convertono i tempi in temperature.

```
void leggi()
{
  String t1;
  String t2;
  long time1;
  long time2;
  int datoDaButtare=0;
  Serial.print("8");
  for(int i=0;i<8;i++)
  {
    t1="";
    t2="";
    for(int j=0;j<4;j++)
      t1+=leggiDato();
    time1=converti(t1);
    datoDaButtare=leggiDato();
    for(int j=0;j<4;j++)
      t2+=leggiDato();
    time2=converti(t2);
    datoDaButtare=leggiDato();
    if(time1==65535 && time2==65535)
      temperature[i]=-100000;
    else
      temperature[i]=(double)time1/(0.0047*
                     (double)time2)-68.09;
  }
  datoDaButtare=leggiDato();
}

char leggiDato()
{
  while(Serial.available()<=0){}
  return (char)Serial.read();
}

long converti (String s)
{
```

```
  double val=0;
  char l='.';
  for(int i=0;i<4;i++)
  {
    l=s[3-i];
    if(l<58 && l>47)
      val+=((long)l-48)*esp[i];
    else
      val+=((long)l-55)*esp[i];
  }
  return val;
}
```

4.4.4 Pagina index.html

Per creare la pagina index.html bisogna definire una funzione come detto nel paragrafo 4.2.5. Per questo è stata creata la funzione tempCmd, dove per prima cosa viene scritta l'intestazione della pagina HTML. Tramite il ciclo for, vengono inserite nella pagina web le temperature dei sensori, la descrizione e la barra. La serie di if in cascata serve appunto per definire colore e lunghezza della barra.

```
void tempCmd(WebServer &server,
WebServer::ConnectionType type, char *, bool)
{

  server.httpSuccess();
  if (type != WebServer::HEAD)
  {

    P(head)="<html><head>"
    "<title>MONITORAGGIO TEMPERATURA CAVI SMART GRID"
    "</title><script>setTimeout(\"location.reload()\""
    ",2000)</script><style type=\"text/css\">"
    "div#contenitore {height:50px;position:relative;"
    "left:10px}div#scritta{position:absolute; height:"
    "25px; width:170px;font-size:16px}div#descrizione"
    "{position:absolute; height:15px;font-size: 13px;"
    "top: 15px}body{margin-top:25px;background-color:"
    "#CCFFFF; line-height: 30px;font-family: arial,"
    " Verdana, sans-serif;}</style></head>"
```

```
"<body  onload=\"doLoad()\">"
"<h1 align=\"center\">MONITORAGGIO TEMPERATURA"
"<br>CAVI SMART GRID</h1><br>";
P(div)="</div><div style=\' position:absolute;"
"height:20px;border:1px solid #000000;left:170px;"
"width:260px; bottom:26px;\'>";
server.printP(head);


for(int i=0;i<8;i++)
{
  server << "<div id=\"contenitore\">";
  server <<"<div id=\"scritta\">";
  server << "sensore "<< i+1 << ": ";
  int intero=0;
  int lunghezza=0;
  int rosso=0;
  int verde=0;
  if(temperature[i]!=-100000)
  {

     if(temperature[i]<soglie[0])
     {
       intero=0;
       verde=255;
       lunghezza=20;
     }
     else if(temperature[i]<soglie[1])
     {
       verde=255;
       intero=(int)((temperature[i]-soglie[0])/
              (soglie[1]-soglie[0])*8);
       rosso=intero*31;
       lunghezza=20+intero*10;
     }
     else if(temperature[i]<soglie[2])
     {
       rosso=255;
       intero=(int)((temperature[i]-soglie[1])/
              (soglie[2]-soglie[1])*8)+8;
```

```
            verde=16*(24-intero)-1;
            lunghezza=20+intero*10;
          }
          else if(temperature[i]<soglie[3])
          {
            rosso=255;
            intero=(int)((temperature[i]-soglie[2])/
                    (soglie[3]-soglie[2])*8)+16;
            verde=16*(24-intero)-1;
            lunghezza=20+intero*10;
          }
          else
          {
            rosso=255;
            lunghezza=260;
          }
          server << temperature[i] << "&ordm;C";
        }
        else
          server << "NC";
        server.printP(div);
        server<<"<div style=\'height:20px; width:"
            <<lunghezza<<"px;background-color:rgb("
            << rosso;
        server<<","<<verde<<",0)\'>"<<"</div></div>";
        server<<"<div id=\"descrizione\">  ";
        for(int h=0;h<40;h++)
        {
          server << descrizioni[i][h];
        }
        server << "</div>";
        server << "</div>";
      }
      server << "<br><ahref=\"config.html\">"
                "Configurazioni</a></body></html>";
    }
}
```

4.4.5 Pagina config.html

La pagina di configurazione è stata chiamata config.html e la relativa funzione è configCmd. Per accedervi bisogna conoscere nome utente e password, infatti è stato creato un controllo grazie al metodo checkCredentials(), che permette di verificare se sono stati inseriti nome utente e password corretti. Nel caso non ci siano parametri passati alla pagina con il metodo POST viene visualizzato il form di modifica dei parametri, in caso contrario si leggono uno ad uno i valori e si salvano in memoria EEPROM. Prima del salvataggio vengono controllati affinché rispettino i vincoli imposti:

- la password deve avere otto caratteri;
- le celle dove inserire l'indirizzo IP devono contenere un numero intero positivo minore di 255 e non possono contenere caratteri diversi da numeri;
- le soglie devono essere numeri interi di tre cifre e non possono contenere caratteri diversi da numeri;
- le descrizioni possono contenere al massimo quaranta caratteri alfanumerici.

Nel caso in cui venga a mancare uno di questi vincoli quel dato non viene salvato in memoria e si riporta nella pagina web l'errore commesso.

```
void configCmd(WebServer &server,
    WebServer::ConnectionType type, char *, bool)
{
  if (server.checkCredentials(pass64))
  {
    if (type == WebServer::POST)
    {
      char name[9], value[40];
      server.readPOSTparam(name, 9, value, 9);
      if(value[7]!=0)
      {
        for(int i=0;i<8;i++)
        {
          EEPROM.write(10+i,value[i]);
          pass[6+i]=value[i];
        }
```

```
  encode(pass,14,pass64,20);
  server<<"password modificata correttamente";
}
else if (value[0]!=0)
  server << "password troppo corta<br>";
int d;
int err=0;
int vuoto=0;
char ipp[12];
for(int i=0;i<4;i++)
{
  d=0;
  server.readPOSTparam(name, 9, value, 9);
  if(value[0]==0 && value[1]==0 && value[2]==0)
    vuoto=1;
  for(int l=0;l<3;l++)
  {
    if(value[l]>47 && value[l]<58)
      d+=(value[l]-48)*molt[l];
    else if(value[l]==0){d/=10;}
    else
      err=1;
  }
  if(d<10)
  {
    ipp[3*i]='0';ipp[3*i+1]='0';
    ipp[3*i+2]=value[0];
  }
  else if(d<100)
  {
    ipp[3*i]='0';ipp[3*i+1]=value[0];
    ipp[3*i+2]=value[1];
  }
  else
  {
    ipp[3*i]=value[0];ipp[3*i+1]=value[1];
    ipp[3*i+2]=value[2];
  }
  if(d>255)
    err=1;
```

```
}
if(err==0 && vuoto==0)
{
  for(int i=0;i<12;i++)
    EEPROM.write(20+i,ipp[i]);
  server << "ip modificato <br>";
}
else if(err==1)
  server << "ip errato<br>";
int errore;
int indice=0;
char soglia[3];
for(int i=0;i<4;i++)
{
  vuoto=0;
  errore=0;
  int s;
  server.readPOSTparam(name, 9, value, 9);
  if(value[0]!=0 || value[1]!=0 || value[2]!=0)
  {
    for(int h=0;h<3;h++)
    {
      if(value[h]!=0 && (value[h]<48 ||
          value[h]>57) )
        errore=1;
    }
    if(errore==1)
      server << "soglia " << i+1<< " errata<br>";
    else
    {
      if(value[2]==0 && value[1]==0)
      {
          soglia[0]='0';soglia[1]='0';
          soglia[2]=value[0];
      }
      else if (value[2]==0)
      {
          soglia[0]='0';soglia[1]=value[0];
          soglia[2]=value[1];
      }
```

```
            else
            {
                soglia[0]=value[0]; soglia[1]=value[1];
                soglia[2]=value[2];
            }
            s=0;
            for(int h=0;h<3;h++)
            {
              s+=(soglia[h]-48)*molt[h];
              EEPROM.write(40+indice+h,soglia[h]);
            }
            soglie[i]=s;
            server<<"soglia "<<i+1<<" modificata<br>";
          }
        }
        indice+=3;
      }
      for(int i=0;i<8;i++)
      {
        server.readPOSTparam(name, 9, value, 40);
        if(value[0]!=0)
        {
          for(int h=0;h<40;h++)
            EEPROM.write(100+50*i+h,value[h]);
          server<<"descrizione "<<i+1
                <<" modificata<br>";
        }
      }
      for(int i=0;i<8;i++)
      {
        for(int j=0;j<40;j++)
          descrizioni[i][j]=EEPROM.read(i*50+j+100);
      }
      return;
    }
    server.httpSuccess();
    if (type != WebServer::HEAD)
    {
        P(message)=
        <<<form HTML>>>;
```

```
                server.printP(message);
    }
  }
  else
  {
    server.httpUnauthorized();
  }
}
```

4.4.6 Pagina sensori.html

La pagina sensori.html serve per fornire le temperature al software MRTG. Questa pagina web necessita di un parametro passato attraverso il metodo GET, il cui nome è "s" e il cui valore deve essere compreso tra 1 e 8. Il contenuto della pagina sarà la temperature relativa al sensore selezionato con il parametro "s". La pagina sensori.html verrà creata utilizzando la funzione sensoriCmd.

```
void sensoriCmd(WebServer &server,
WebServer::ConnectionType type, char *url_tail, bool)
{
  if (type == WebServer::GET)
  {
    char name[3];
    char value[3];
    if (strlen(url_tail))
    {
      int rc = server.nextURLparam(&url_tail,
                          name, 3, value, 3);
      if(name[0]=='s'&&value[0]>48&&value[0]<57)
      {
        if(temperature[value[0]-49]==-100000)
          server << "NC";
        else
          server << temperature[value[0]-49];
      }
    }
  }
}
```

4.4.7 Funzione setup

Come detto nel capitolo 4.1 la funzione setup() viene invocata una volta all'avvio della scheda Arduino, per questo motivo all'interno di questa funzione si legge lo stato del pulsante di reset. Se questo è premuto o se non vi sono ancora parametri salvati all'interno della EEPROM si procede caricando quelli di default. Viceversa si leggono i parametri salvati nella memoria EEPROM e si scrivono all'interno delle relative variabili. Si procede quindi con l'inizializzazione della seriale, del modulo ethernet e del web server.

```
void setup()
{
  pinMode(8, INPUT);
  int buttonState = digitalRead(8);
  String ipep="192168001003";
  String soglieep="015025030040";
  byte ip[] ={ 192,168,1, 3 };
  soglie[0]=15;
  soglie[1]=25;
  soglie[2]=30;
  soglie[3]=40;
  char controllo=(char)EEPROM.read(0);
  if(controllo=='c' && buttonState!=HIGH)
  {
    ipep="";
    soglieep="";
    for(int i=0;i<8;i++)
      pass[6+i]=(char)EEPROM.read(10+i);
    for(int i=0;i<12;i++)
      ipep+=(char)EEPROM.read(20+i);
    for(int i=0;i<12;i++)
      soglieep+=(char)EEPROM.read(40+i);
    encode(pass,14,pass64,20);
    for(int i=0;i<8;i++)
    {
      for(int j=0;j<40;j++)
        descrizioni[i][j]=EEPROM.read(i*50+j+100);
    }
  }
  else
  {
```

```
      for(int i=0;i<8;i++)
        EEPROM.write(10+i,pass[6+i]);
      for(int i=0;i<12;i++)
        EEPROM.write(20+i,ipep.charAt(i));
      for(int i=0;i<12;i++)
        EEPROM.write(40+i,soglieep.charAt(i));
      for(int i=0;i<8;i++)
      {
         for(int j=0;j<40;j++)
         {
            descrizioni[i][j]=0;
           EEPROM.write(100+i*50+j,0);
         }
      }
     EEPROM.write(0,'c');
   }
   for(int i=0;i<4;i++)
       ip[i]=(ipep.charAt(i*3)-48)*100+
         (ipep.charAt(i*3+1)-48)*10+
         (ipep.charAt(i*3+2)-48);
   for(int i=0;i<4;i++)
       soglie[i]=(soglieep.charAt(i*3)-48)*100+
         (soglieep.charAt(i*3+1)-48)*10+
         (soglieep.charAt(i*3+2)-48);
   Ethernet.begin(mac, ip);
   Serial.begin(57600);
   leggi();
   webserver.setDefaultCommand(&tempCmd);
   webserver.addCommand("index.html", &tempCmd);
   webserver.addCommand("config.html", &configCmd);
   webserver.addCommand("sensori.html", &sensoriCmd);
   webserver.begin();
}
```

4.4.8 Funzione loop

Nella funzione loop(), che verrà chiamata ripetutamente, si leggono ogni secondo le temperature dai sensori, richiamando la funzione leggi(). A ogni ciclo, inoltre, se ci sono connessioni entrati verranno servite grazie alla funzione processConnection.

```
void loop()
{
  if((millis()-tempoRilettura)>1000)
  {
    leggi();
    tempoRilettura=millis();
  }
  char buff[64];
  int len = 64;
  webserver.processConnection(buff, &len);
}
```

Capitolo 5

Conclusioni

In conclusione di questo elaborato si riporta il risultato finale dell'installazione. La posa in opera è stata seguita dai tecnici del laboratorio e dai servizi tecnici dipartimentali. E' stato posato un cavo ethernet che dal piano terra (quadro smart grid del laboratorio PEL) arriva al sottotetto in cui sono presenti i cavi. Oltre ai dati, tale cavo porta anche l'alimentazione del sistema.
Nelle figure sottostanti si osserva l'applicazione dei sensori sui relativi cavi. Il sensore n. 1 è posizionato in modo da rilevare la temperatura ambiente.

Figura 5.1: foto del sistema installato nel sottotetto

Infine, si riporta lo screenshot (figura 5.3) della pagina web con le temperature e descrizione della locazione del sensore. Questa pagina è accessibile all'indirizzo ip 147.162.11.8.

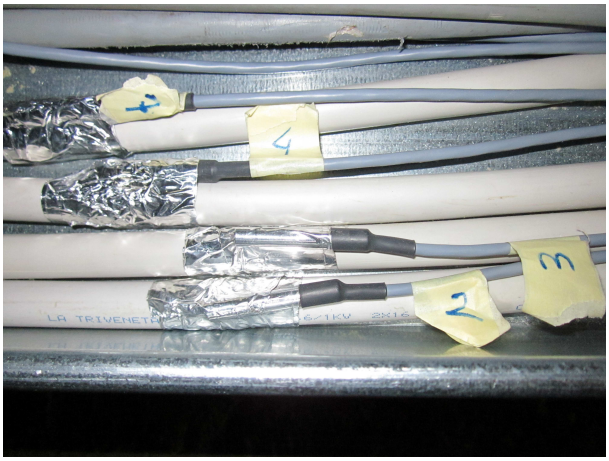


Figura 5.2: foto dei sensori appaiati ai cavi

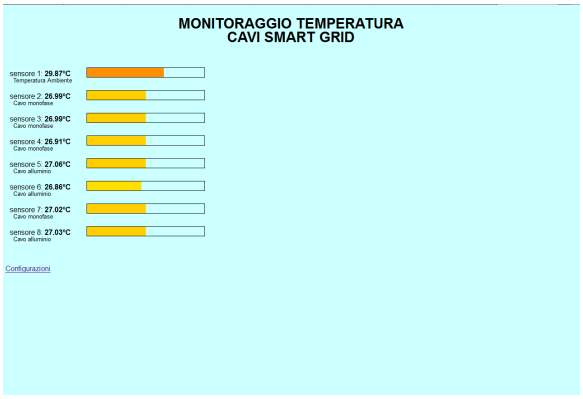


Figura 5.3: pagina iniziale

Il presente lavoro di tesi ha portato alla realizzazione di un sistema che consente di monitorare in tempo reale eventuali anomalie dovute al surriscaldamento dei cavi del progetto smart grid.

Manuale utente

- OTTO SENSORI DI TEMPERATURA
- RANGE DI TEMPERATURE: -45°C +130°C
- PORTA ETHERNET
- ALIMENTAZIONE TRA 7 E 12V
- VISUALIZZAZIONE DELLE TEMPERATURE ATTRAVERSO PAGINE WEB

PARAMETRI INIZIALI:
IP: 192.168.1.3
NOME UTENTE: admin
PASSWORD: password

Introduzione

Questo sistema permette l'acquisizione delle temperature e la relativa visualizzazione attraverso delle pagine web in forma testuale e grafica. Connettendolo ad una rete LAN e alimentandolo attraverso un alimentatore esterno, il sistema sarà subito funzionante.

Connessioni e impostazioni iniziali

Alimentare il sistema attraverso il connettore industriale a due vie (ALIMENTAZIONE COMPRESA TRA 7V e 18V) e connettere la porta ethernet con quella del pc attraverso un cavo di rete. Impostare l'IP del computer con un indirizzo del tipo 192.168.1.X dove X deve essere un numero compreso tra 1 e 254 diverso da 3. Ora, aprire un browser web, ad esempio Mozzilla Firefox, e inserire nella barra degli indirizzi l'IP 192.168.1.3. Così facendo si aprirà la pagina dove vengono visualizzate le temperature (fig

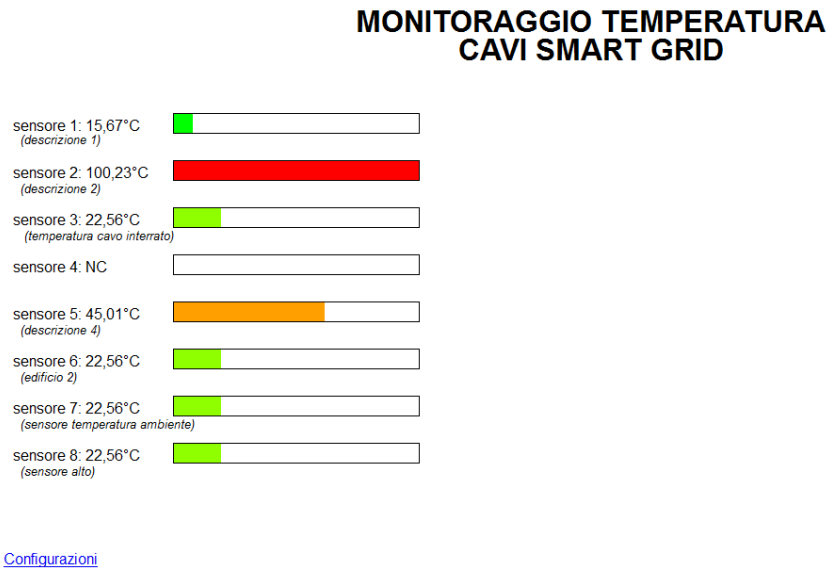


Figura 4: pagina iniziale

4).
Per modificare le impostazioni iniziali cliccare sul link “Configurazioni”, in maniera tale da accedere alla pagina di configurazione (fig 5), dove si possono modificare l’indirizzo IP, password, soglie e descrizioni dei sensori. Per rendere definitiva la modifica dell’IP bisogna, inoltre, riavviare il sistema di acquisizione scollegando e ricollegando l’alimentazione. Questo passaggio va eseguito solo per l’IP, gli altri parametri si aggiornano automaticamente.
Modificato l’indirizzo IP, in modo tale da poterlo inserire all’interno della propria rete LAN, il sistema è pronto per l’utilizzo e vi si potrà accedere da qualsiasi pc connesso alla rete.

Configurazioni

Grazie alla pagina i configurazione sarà possibile modificare:

- password della pagina privata, contenente otto caratteri;
- IP del sistema di monitoraggio;
- le soglie, sono dei valori che regolano l’avanzamento delle barre colorate nella pagina iniziale;

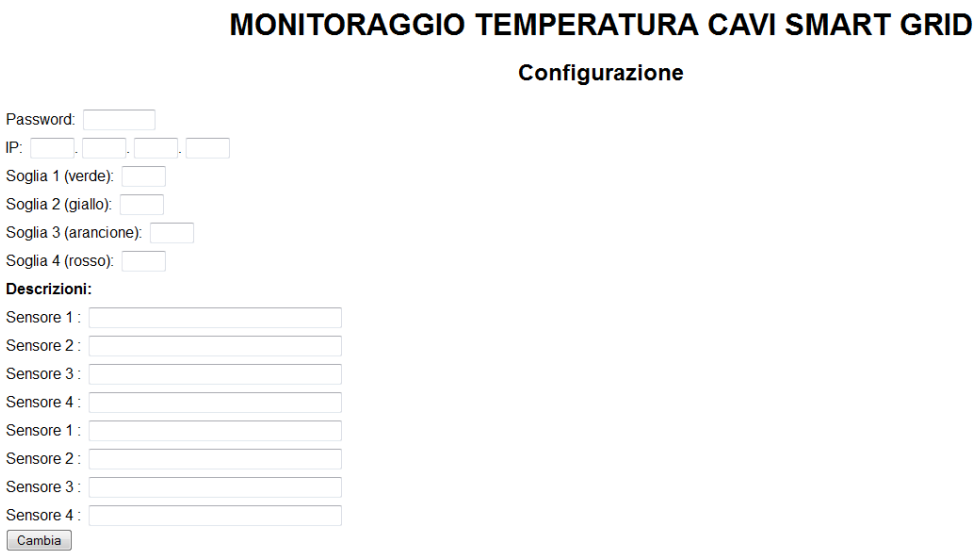


Figura 5: pagina di configurazione

  - VERDE: temperatura sotto controllo;
  - GIALLO: temperatura appena oltre il valore normale;
  - ARANCIONE: temperatura critica;
  - ROSSO: temperatura fuori controllo.

- descrizioni dei sensori di temperatura.

Caricare programma su Arduino ethernet

Per iniziare si deve scaricare l'ambiente di sviluppo di Arduino (ver. 1.0), la libreria Webduino (ver. 1.7), reperibile all'indirizzo:
https://github.com/sirleech/Webduino , e il file "board.txt" per l'ambiente di sviluppo versione 1.0, scaricabile sul sito arduino.cc nella pagina riguardante l'Arduino ethernet board.
DECOMPRIMERE il file contenente l'ambiente di sviluppo Arduino e, copiare all'interno della cartella "libraries" la libreria Webduino.
Il file "board.txt", invece, deve essere copiato all'interno della cartella "hardware/arduino". Ora, scaricare dal sito di Arduino i driver per l'USB Serial Light Adapter (adattatore USB seriale di Arduino) e scompattarli all'interno della cartella "drivers".
Prima di connettere l'Arduino al pc bisogna assicurarsi che non vi sia

niente connesso alla scheda, quindi scollegare alimentazioni esterne, cavi connessi alla seriale o su altri pin. Collegare, ora, Arduino all'adattatore seriale, utilizzando i pin sul lato più corto di Arduino ethernet (vedi fig 6) e l'adattatore al pc tramite un cavo USB. Una volta collegato attendere

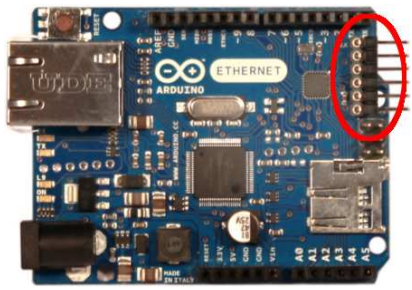


Figura 6: Pin per il collegamento con l'adattatore seriale

fino a quanto il pc non chiederà i driver, selezionare la cartella "drivers", dove è stato scaricato il driver per l'adattatore seriale, e procedete con l'installazione. Avviare l'ambiente di sviluppo Arduino, aprire il file da caricare, selezionare dal menù Tools/board la scheda Arduino ethernet e dal menù Serial Port la porta seriale dove è connesso (normalmente è già selezionata), infine premere il pulsante Upload situato nella barra degli strumenti.
Attendere la compilazione e poi il caricamento, infine, scollegare l'adattatore USB seriale e riconnettere ad Arduino l'alimentazione esterna e le connessioni con altri dispositivi.

Appendice A

Datasheet

Di seguito vengono riportati i datasheet del sistema di acquisizione Smartec SMTA08 e dei sensori di temperatura Smartec SMT16030.

SMart

Temperature

Acquisition

System

for

08 channels

SMTAS08

Contents.

1. Introduction

This document describes an 8 channel temperature measurement system: The Smart Temperature Acquisition System(SMTAS08). It is based on the use of the Smart-temperature sensors SMT160 of Smartec. The SMT160 is a three-terminal integrated temperature sensor with a duty cycle output. Two terminals are used for the power supply of 5 volts and the third terminal carries the output signal. The output signal of the sensor is a duty-cycle-modulated square-wave signal (see Figure 1).

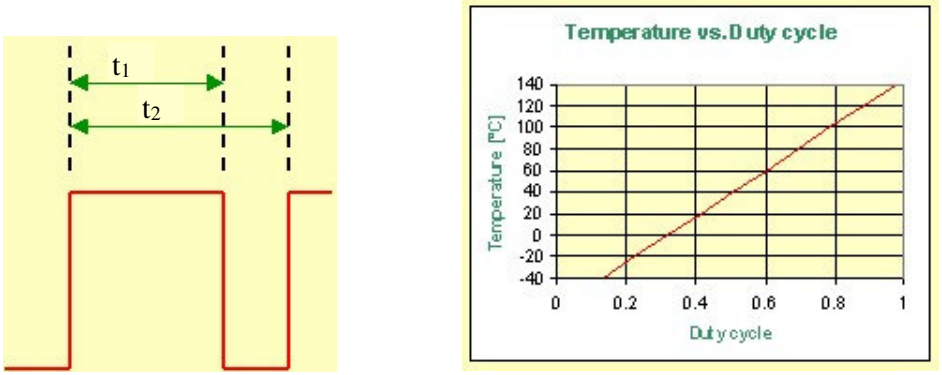


Figure 1 (a) Output signal of the SMT, (b) Relation between the duty-cycle and the temperature.

After measuring both t_1 and t_2, the temperature in ^{0}C can be calculated by equation:

$$\theta = \frac{t_1}{0.0047 t_2} - 68.09$$

The temperature sensors are sold separately from the SMTAS08 board, because the SMT 160-30 temperature sensor is available in different encapsulations (TO18,TO92, TO220, etc), each with their specific properties. One important issue is their accuracy. The TO18 version yields the most accurate sensor and has an accuracy of 0.7 ^{0}C. The complete specification of the temperature sensor range is presented in the datasheet, which should be consulted in conjunction with this document.

The SMTAS08 is equipped with a microcontroller of the type Microchip PIC16F876. An RS232 interface chip offers external serial communication with the microcontroller. In turn each of the sensors is powered by a corresponding processor output. An analog multiplexer (74HC4051), which is controlled by the microcontroller, selects one of the eight sensor outputs by connecting it to the microcontroller input port. The input is then sampled to measure the values of *t1* and *t2* for this sensor. This process is repeated for each of the sensors, after which the 8 values of *t1* and *t2* are sent to the RS232 port.
A block diagram and a photograph of the system are shown in Figure 2 and Figure 3, respectively.

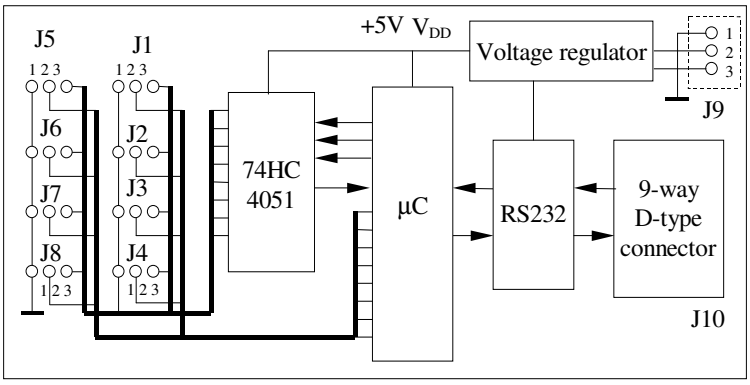


Figure 2 Functional block diagram.

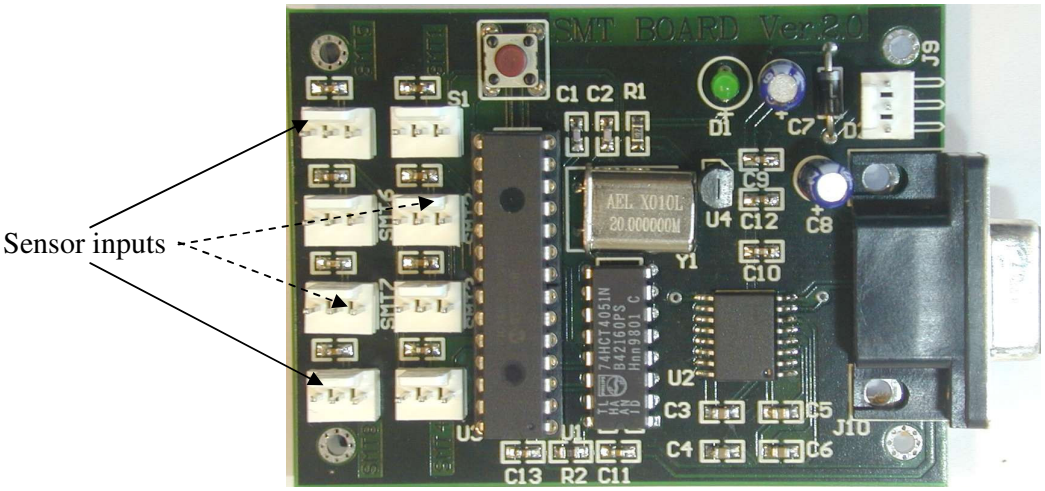


Figure 3 A photograph of the SMTAS08 system.

2. Quick start and functional check

This document assumes that the SMTAS08 board has to be connected to a personal computer or a laptop. Any other device capable of handling RS232 data could do the job just as well however.
In order to get the board running, a few things have to be prepared. Please find the hardware checklist below:

- The SMTAS08 board itself
- One or more SMT 160-30 sensors
- A D9 RS232 connecting cable (straight)
- A power supply delivering between 7 and 18 Volts (min 20 mA)
- A PC or laptop running a terminal program for instance Windows Hyper terminal (57600,8,none,1,none)

After connecting the parts together temperatures can be measured in case one or more temperature sensors are connected to the SMTAS08 board. Make sure the com port parameters, as mentioned above, and the portnumber itself are chosen correctly. Type “m” and the datastream should start flowing. More information about this will be given later but this is the fastest way to check whether the SMTAS board is functioning correctly.

Please refer to section 4 for the pin layout of the board and to the datasheet of the SMT160-30 for the sensor pin connections. In case no sensors are connected to the SMTAS08 board all data will appear as “FFFF”.

3. Inside the SMTAS08 system

General

The SMTAS08 system can be used to measure up to 8 Smartec temperature sensors. The microcontroller measures the output signal of the selected sensor, provides the 5 V power supply for the selected sensor and communicates with the outside digital world. Figure 4 shows the flowchart of the program in the microcontroller.

For speed of measurement of the duty-cycle of the sensors, the program of the PIC processor is written in Assembler. The processor is mounted on a socket so the user is free to write his own program if desired.

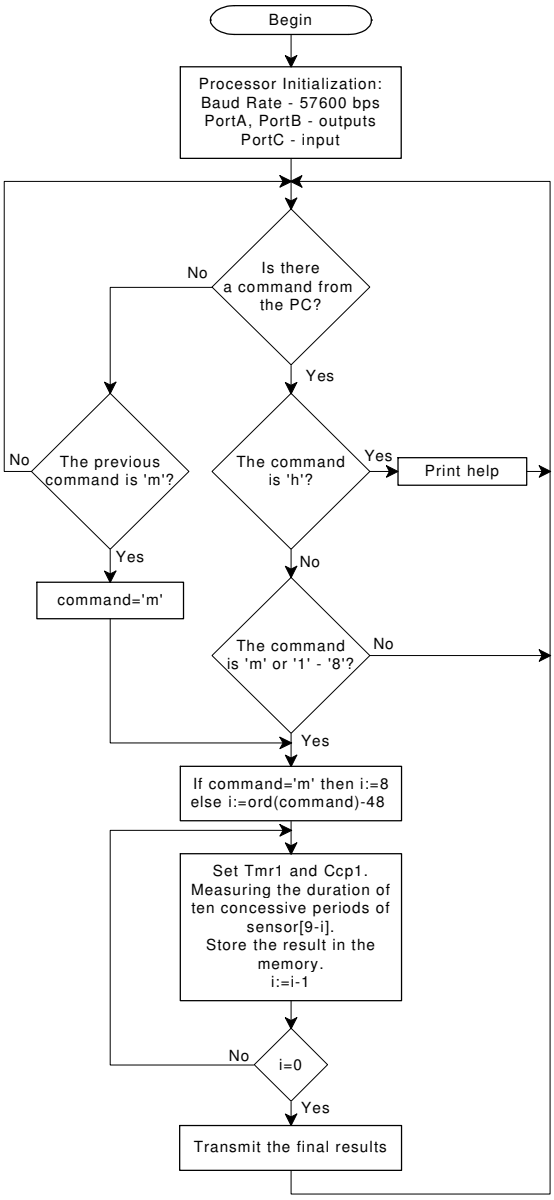


Figure 4 Flowchart of the program in the microcontroller.

Self-heating of temperature sensor

Each sensor needs some power, however little, which leads to an error in the temperature measurement. By switching off the power of the sensors that are not being sampled and by making sure the measurement duration of each sensor is as short as possible the effect of selfheating is very small. It is even smaller than the resolution of the sensor; so it can be neglected.
Each of the eight sensors is only switched on for a period of maximum 10.5 ms. In figure 5 the timing diagram of the subsequent sensors is depicted. With all sensors mounted, each sensor is being powered about 10% of time and when fewer sensors are mounted this percentage will increase towards 35% (one sensor is mounted).

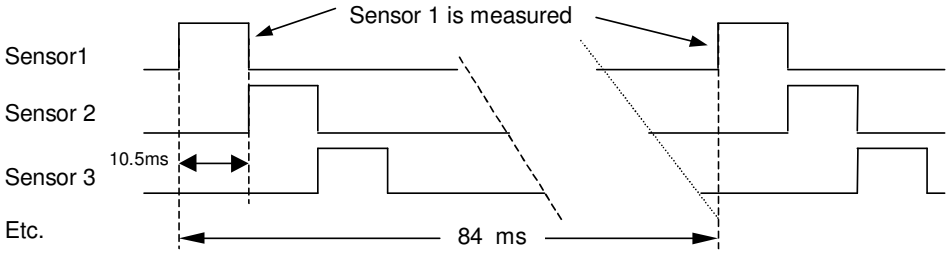


Figure 5 Timing diagram of subsequent sensors. (times given in timing-diagram are typical)

Measurement accuracy

The resolution (i.e. the accuray with which the sensor output duty cycle is determined), depends on the number of samples taken. With the choosen sampling speed (5 MHz) and duration (about 10.5 ms) a resolution of 0.015 °C is obtained and at the same time the self heating effect is minimized. The absolute accuracy of the all over measurement system still depends on the sensors (0,7 °C at best for the TO18).

3. Circuit diagram and connections to the board

General

In figure 6 the circuit diagram of the SMTAS08 is given. The on-board voltage regulator provides the internal power supply voltage of 5 V. The SMTAS08 board requires a DC supply voltage in the range of 7 V to 18 V. The on-board LED will light when the supply voltage is connected.
If needed a reset switch can be mounted (SW DIP-2). This may be convenient when developing software by yourself.

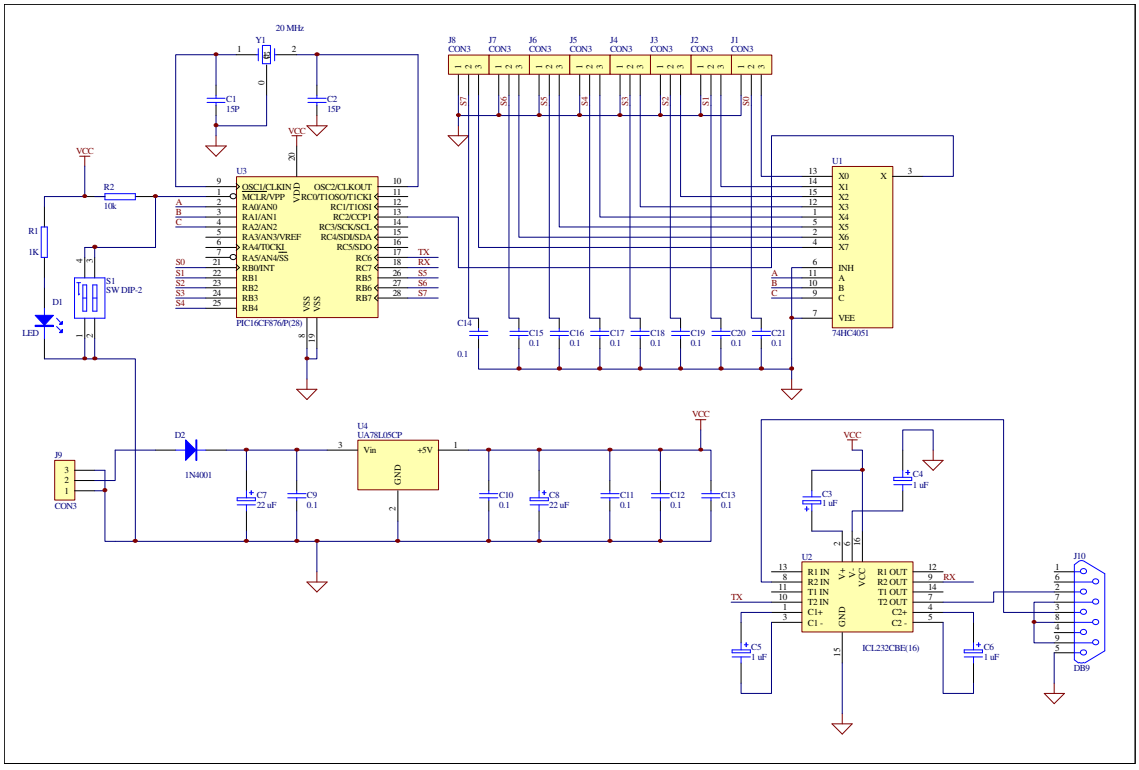


Figure 6 Circuit diagram of the SMTAS08 system.

Connector layout

Connection to the SMTAS08 system board is implemented with 10 connectors:
J1 ~ J8 3-pin header to connect 8 sensors.
J9 3-pin header for power supply.
J10 9-way D-type connector for communication with a PC (standard serial cable, straight)

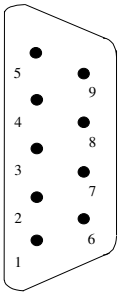
Figures 7, 8 and 9 show the pin connection of connectors.

1	1	Ground
2	2	Power supply to sensor
3	3	Output of sensor

Figure 7. The pin configuration of J1 ~ J8 (pitch 0.1 “).

1	1	Ground
2	2	External power supply of the board 7 – 15 volt DC
3	3	Ground

Figure 8. The pin configuration J9 (pitch 0.1”).



1	NC	Not connected
2	T1	Serial data from PC to board
3	R1	Serial data from board to PC
4	NC	Not connected
5	GND	Ground
6	NC	Not connected
7	NC	Not connected
8	NC	Not connected
9	NC	Not connected

Figure 9. The pin configuration of J10 (9 pole SUB-D female).

5. SMTAS8 software

There are many ways to display results from the system. As an example, we will discuss a terminal program under Windows(**Hyper terminal**), and our executable **LABVIEW**. Finally, an example in **Visual Basic** is available.

Hyper terminal

Configuration

A terminal program fi. Hyper terminal under Windows can easily display the measurement results via the serial port of the PC. The configuration is: speed 57600 bps, 8 data bit, 1 stop bit and No parity.
Once the communication between the microcontroller and the PC is established, you can get help infromation by typing “h”. The following will be displayed:

```
S M T A S 8  T E M P E R A T U R E
M E A S U R E M E N T  S Y S T E M
     Version :

m -> Infinite measurement cycle
1 -> First sensor, single measurement
2 -> First two sensors, single measurement
3 -> First three sensors, single measurement
4 -> First four sensors, single measurement
5 -> First five sensors, single measurement
6 -> First six sensors, single measurement
7 -> First seven sensors, single measurement
8 -> First eight sensors, single measurement
s -> Stop

Please Make a Choice :
```

As soon as you press "m" or a digit between "1" and "8", the board will start sending data. The data format is Hexadecimal. For each sensor, both t_1 and t_2 are represented by four hexadecimal Ascii digits, separated by " " (space) and followed by "/n" or "CRLF" at the end. In case that there is no sensor connected to a selected position, or when a selected sensor does not transmit a period-modulated output signal, this will be detected by the microcontroller, which will send "FFFF" for both *t1* and *t2*.

Below you find a typical example of a terminal output. As you can see, sensor 8 is not connected.

```
1B36 3F92 1AEF  3ED2  1A6B 3D95 1AFA 3EC3  1BFB 413A 1B08 3EF9 19FB 3C7C  FFFF FFFF
1B34 3F89 1AF1  3ED3  1A65  3D8F 1AFB 3ECD 1BFA 4138 1B13 3F14  19FA 3C7E  FFFF FFFF
1B34 3F8E 1AED 3ECD 1A68  3D91 1AF6  3EC2  1BF7  4130 1B15 3F15  19F6  3C7A FFFF FFFF
```

Sensor 1 Sensor 2 Sensor 3 Sensor 4 Sensor 5 Sensor 6 Sensor 7 Sensor 8

Figure 10. Typical example of a Terminal program output (Sensor 8 not connected).

To obtain the corresponding temperature, the values for t_1 and t_2 have to substituted in formula (1). A terminal program cannot do this calculation, but gives you a convenient way of displaying the board data. Included with the board is also a Labview program, which will do the calculations as well and display the actual temperatures.

Executable LabView

A copy of the latest version of the SMTAS08 labview executable can be found at www.smartec.nl/supportshop. Two versions are available; one in ZIP-format and one in ARJ-format. Both versions contain SMTAS08.exe, which is the actual program and SERPDRV which is the serial portdriver needed by labview. The port driver needs to be located in the same directory as the executable. The program can be started by double clicking the SmTAS08.exe. When the program starts the labview window appears. There is one output mode selection item and there are four control items: ***NumSensor, Serial port No. Offset correction*** and ***Average Index***. The display items are: ***Real-time value***, ***Std Deviation*** and ***Average value***.

Output mode selection

Using the output mode selection, ***OUTPUT MODE***, the measured temperatures can be displayed in Celcius, Fahrenheit or Kelvin.

Control items

The item ***Serial port No***.. allows the user to select the desired serial port of the PC.
The item ***NumSensor*** is used to select the number of sensor to be measured (maximum is 8).
The item ***Offset Correction*** is used to correct the tolerance of the sensors. For instance, to use 8 sensors to measure a temperature, there is a deviation from sensor to sensor due to the fabrication tolerance. Then, the average value of these 8 sensors at a certain same temperature is used to correct this deviation.
The item ***Average Index*** is used to set the number of measurements to be averaged. A large number of measurements results in lower noise, but also increases the measurement time.

Display items

The item ***Real-time value*** enables the numerical display of the measurement result. Meanwhile, a graphic chart displays the real-time value by a red line.

The item ***Average Value*** averaging over a variable number of measurements indicated by the ***Average Index***.
The item ***Std Deviation*** indicates the standard deviation of the system for a variable number (indicated by ***Average Index***) of measurements.

To Start Measurement

Double click the button ⇒ at the top-left corner of the front-panel window to start the measurement. After a few seconds the results are displayed at the monitor.

Visual Basic.

To be implemented

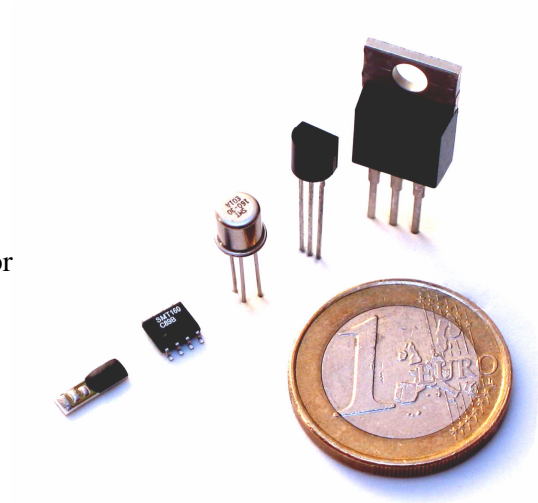
6. Ordering information

SMTAS08	Temperature Acquisition System for 8 Smartec temperature sensors.
SMT1603018	Smartec temperature sensor in TO18 encapsulation (metal can)
SMT1603092	Smartec temperature sensor in TO92 encapsulation (commercial)
SMT16030220	Smartec temperature sensor in TO220 encapsulation
SMT16030SO	Smartec temperature sensor in SOIC8
SMT16030HE	Smartec temperature sensor as small hybrid (2.5. x 8 mm)

SMT16030 DIGITAL TEMPERATURE SENSOR

Features

Absolute accuracy ± 0.7 °C
Linear output within 0.2 °C
Resolution better then 0.005 °C
Duty Cycle output
Calibrated on chip
TTL, CMOS compatible
Temperature range 175 °C (-45 to +130 °C)
Directly connectable to data input of microprocessor
Easy multiplexing of multiple sensors



Typical applications

Heater systems
Measuring instruments
Washing machines
Overheating protection
Appliances

Introduction

The Smartec temperature sensor is a sophisticated full silicon temperature sensor with a digital output. The one wire output (duty-cycle modulated) can be directly connected to all kinds of micro-controllers without the need of A/D conversion. The temperature range is –45 °C to 150 °C. The high resolution (< 0.005 °C) makes the sensor useful for high precision applications. The sensor is available in various housings like T018, T092, T0220 and for high volume production in SOIC. Special housing can be manufactured on request.

Product highlights

The SMART TEMPERATURE SENSOR features a duty-cycle modulated square wave output voltage with linear response to temperatures in the -45 °C to +130 °C range. The absolute accuracy is better than 1.2 °C. In the range from -30 to +100 °C absolute accuracy is better than 0.7 °C, while the linearity is better than 0.2 °C (Model TO18).

SMARTEC B.V, Delpratsingel 26
4811 AP Breda, The Netherlands
WWW.SMARTEC.NL
Tel: +31(0)765205353

The SMART TEMPERATURE SENSOR is calibrated during test and burn-in of the chip. The integrated modulator ensures the sensor unit can communicate effectively with low-cost microcontrollers without the need of (onboard) A/D converters or an Xtal controlled oscillator.

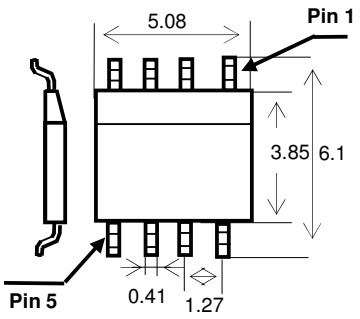
The SMART TEMPERATURE SENSOR combines digital output and on-chip calibration to ensure major cost reductions and performance-related advantages.

In applications where multiple sensors are used, easy multiplexing can be obtained by using a corresponding number of microprocessor inputs or by using low cost digital multiplexers.

Since the sensor requires no subsequent calibration, optimal cost savings are recorded both during manufacturing and in the course of after-sales servicing.

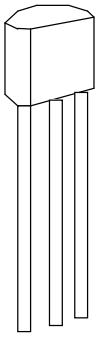
Pin-out and housing.

SOIC-8



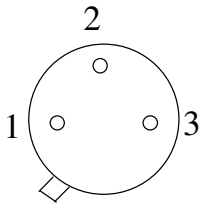
Pin 1 +V
Pin 7 GND
Pin 8 Out
All Sizes in mm.

TO92



1 2 3

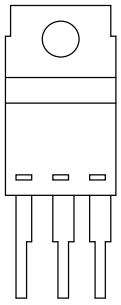
TO18



bottom view

1 Output
2 + Vcc
3 GND

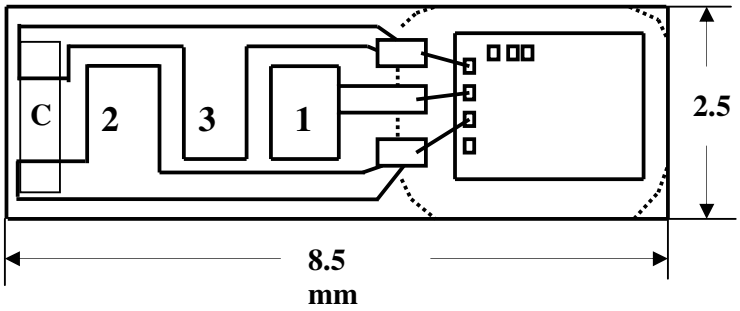
TO220



1 3 2

metal backplate = GND

HEC housing



C = 100 nF

(between Vcc and GND)

Pin 1 Output
Pin 2 + Vcc
Pin 3 GND

SMARTEC B.V, Delpratsingel 26
4811 AP Breda, The Netherlands
WWW.SMARTEC.NL
Tel: +31(0)765205353

Specifications

	TO18			TO92	TO220	HEC	SOIC-8	
Parameters	**min**	**typ**	**max**	**max** [1]	**Max**	**max**	**max**	**Units**
Supply voltage[2]	4.75	5	7.2	*	*	*	*	**V.**
Supply current	160		200	*	*	*	*	**µA.**
temperature range[3]	-45	-	130	*	*	*	*	**°C**
Total accuracy[4]	-30 + 100 °C		0.7	1.2	1.7	1.5	1	**°C**
	-45 + 130 °C		1.2	2	1.7	1.5	1.5	**°C**
Non linearity[5]			0.2	0.4	0.5	1.0	1.0	**°C**
Supply voltage sensitivity			0.1	*	*	*	*	**°C/V**
Repeatability			0.1	0.2	0.2	0.2	0.05	**°C**
Long term Drift			0.05	-	-	-	0.05	**°C**
Operating temperature	-45		130	*	*	*	*	**°C**
Storage temperature	-50		150	*	*	*	*	**°C**

Output

- duty cycle	=0.320+0.00470*t (t=temperature in C)							
- frequency	1	-	4	*	*	*	*	**Khz**
- noise			0.005	*	*	*	*	**°C**
- impedance			200	*	*	*	*	**Ohm**
- short circuit	infinite maximum current applied 40 mA							

1 * All not mentioned specifications are the same as for TO18
2 Case connected to ground
3 The SMT 30-160-18 can be used from -65 to +160 °C for a short period without physical damage to the device. The specified accuracy applies only to the rated performance temperature range.
4 Total accuracy includes all errors.
5 Applicable from -30 to +100 °C

SMARTEC B.V, Delpratsingel 26
4811 AP Breda, The Netherlands
WWW.SMARTEC.NL
Tel: +31(0)765205353

Product description

The SMT160-30 is a three terminal integrated temperature sensor, with a duty-cycle output. Two terminals are used for the power supply of 5 Volts and the third terminal carries the output signal. A duty cycle modulated output is used because this output is interpretable by a micro-processor without A-D converter, while the analogue information is still available. The SMT160-30 (TO18 model) has an overall accuracy of 0.7 °C in the range from -30 C to +100 °C and an accuracy of 1.2 °C from -45 to +130 °C. This makes the sensor especially useful in all applications where "human" (climate control, food processing etc.) conditions are to be controlled. Due to its very high resolution (< 0.005K) this sensor is especially suited for applications where very accurate measurements are needed.

The CMOS output of the sensor can handle cable length up to 20 meters. This makes the SMT160-30 very useful in remote sensing and control applications.

Understanding the specifications.

It is important to understand the meaning of the various specifications and their effects on accuracy. The SMT160-30 is basically a bipolar temperature sensor, with accurate electronics to convert the sensor signal into a duty cycle. During production the devices are calibrated.

The output signal

The output is a square wave with a well-defined temperature-dependent duty cycle. The duty cycle of the output signal is linearly related to the temperature according to the equation:

$$D.C. = 0.320+0.00470*t$$

D.C. = duty cycle
t = Temperature in °C

A simple calculation shows that - for instance- at 0 °C:

D.C.= 0.320 or 32.0 % and at 130 °C

D.C.= 0.931 or 93.1 %

In the output frequency of the sensor there is no temperature information, only the duty cycle contains temperature information in accordance to the formula given above. The output signal may show low frequency jitter or drift. Therefore most oscilloscopes and counters are not suited for verifying the accuracy of these sensors. The temperature information contained in the duty-cycle value, however, is guaranteed to be accurate within the values specified for each model (housing).

SMARTEC B.V, Delpratsingel 26
4811 AP Breda, The Netherlands
WWW.SMARTEC.NL
Tel: +31(0)765205353

Total accuracy

The mentioned equation is the nominal one. The maximum deviation from the nominal equation is defined as total accuracy. With temperatures above 100 °C the accuracy decreases.

Non linearity

Non-linearity as it applies to the SMT160-30 is the deviation from the best-fit straight line over the whole temperature range. For the temperature range of -30 °C to +100 °C the non-linearity is less than 0.2 °C (TO18).

Long-term drift

This drift strongly depends on the operating condition. At room temperature the drift is very low (< 0.05 °C). However at higher temperatures the drift will be worse, mainly because of changes in mechanical stress. This drift is partly irreversible and causes non-ideal repeatability and long-term effects. At temperatures above 100 °C (but in the operating range) a long-term drift better than 0.1 °C is to be expected.

Noise

The resolution is better than 0.005 °C. The standard deviation of the noise level (measured over a 20 ms. period) is below this 0.005 °C.

Time constants

The time constant of the sensor is measured under different circumstances.

To compare this with other types of sensors the same kind of measurements were done. The time constant is defined as the time required to reach 63% of an instantaneous temperature change.

The figures mentioned below are difficult to measure; an accuracy of around 5 % is a reasonable estimation. These figures only apply to the sensor built in a TO-18 housing and not the TO-92, the TO220 nor the naked chip. The values found only depend on the physical parameters of the measurement setup.

SMARTEC B.V, Delpratsingel 26
4811 AP Breda, The Netherlands
WWW.SMARTEC.NL
Tel: +31(0)765205353

condition	timeconstant (s) (TO18)
mounted in an alu block of a certain temperature (mean value of different measurements)	0.6
in a bath filled with oil that was stirred (mean value of different measurements)	1.4
Moving air with a speed of about 3 m/s	
- without heatsink	13.5
- with heatsink	5
Non moving air	
-without heatsink	60
-with heatsink	100

Overview of time constants in different conditions

General operation

A simple way of measuring a duty cycle is to use a microcontroller. The sensor output can be directly connected to a microcontroller input. The microcontroller can determine the duty-cycle value by sampling the sensor output. If the microcontroller is not fast enough to determine the temperature accurately enough within one sensor output cycle, the sampling can be extended over multiple periods. This method has the advantage to filter out noise. From the theory of signal processing it can be derived that there is a fixed ratio between the sensors signal frequency, the sampling rate and the sampling noise. This sampling noise limits the accuracy and amounts to:

$$T_{error} = 200 * t_s / sqrt(6 * t_m * t_p)$$

T_{error} = measurement error (= standard deviation of the sampling noise)
t_s = microcontrollers sampling rate
t_m = total measurement time
t_p = output signal periodicity of the sensor

Microcontrollers can sample at a high frequency so with a simple program it is possible to measure the sensor's duty cycle within 50 ms and a resolution of .01 °C.

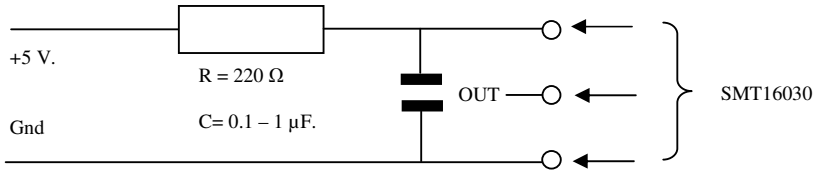
SMARTEC B.V, Delpratsingel 26
4811 AP Breda, The Netherlands
WWW.SMARTEC.NL
Tel: +31(0)765205353

NOTE:

The above mentioned error is NOT related to the intrinsic accuracy of the sensor. It just tells us what happens to the accuracy (standard deviation) of the measurement of a digital signal, when that signal is being sampled by a microcontroller.

About noise protection and how to prevent damage caused by a wrong power supply polarity.

The Smartec SMT16030 is based on a free running oscillator. Periodic spikes on the power supply line may make the oscillator synchronise, resulting in a false temperature reading. To overcome this problem it is advised to put a filter in the power supply line of the sensor. It is suggested to use a low pass RC filter as given below. An additional advantage also is the power supply polarity protection of the sensor. The resistor of 220 Ohm limits the current through the sensor to about 25 mA. At this current the sensor will survive a possible wrong power supply polarity. The software can detect the presence of the output signal and therefore a proper connection of the sensor. See below for a suggested diagram.



Power line noise filtering and polarity damage protection

For more information about how to measure duty cycles by means of a microcontroller, please refer to our application notes, available for download at our website WWW.SMARTEC.NL.

SMARTEC B.V, Delpratsingel 26
4811 AP Breda, The Netherlands
WWW.SMARTEC.NL
Tel: +31(0)765205353