

# *EH – 1*

## *Progetto di un sistema domotico interattivo*



*Sviluppato da  
Marco Di Goro  
VB ELT a.s. 2007/08  
I.T.I.S. "A. Meucci" Firenze*

# Capitolo 1: Introduzione

## Il “significato” del progetto

La domotica è la disciplina che si occupa di studiare le tecnologie atte a migliorare la qualità della vita nell'ambiente domestico grazie all'automazione, al controllo dei processi gestionali e all'integrazione dei sistemi. Il termine "domotica" è infatti un neologismo derivante dalla contrazione della parola latina domus (casa, abitazione) unita al sostantivo "automatica", che quindi indica la "scienza dell'automazione delle abitazioni". Ha dunque come oggetto di studio privilegiato proprio l'automazione della casa. Tutto ciò si ottiene utilizzando in modo combinato elettronica, informatica e meccanica.

Questa disciplina è nata nel corso della terza rivoluzione industriale e si prefigge obiettivi precisi:

- \* migliorare la qualità della vita;
- \* migliorare la sicurezza sotto tutti i punti di vista;
- \* rendere più agevole la vita quotidiana per i portatori di handicap;
- \* risparmiare energia;
- \* semplificare la progettazione, l'installazione, la manutenzione e l'utilizzo della tecnologia;
- \* ridurre i costi di gestione;
- \* convertire i vecchi ambienti e i vecchi impianti.

## Il concetto di casa intelligente

Con l'espressione "casa intelligente" si definisce l'integrazione di diversi dispositivi per il controllo automatizzato di apparati domestici, di sensori di rilevazione dello stato dell'ambiente, di funzioni intelligenti di supporto e di sistemi telecomunicativi.

Un'abitazione così integrata può essere controllata dall'utilizzatore tramite opportune interfacce utente (come pulsanti, telecomando, touch screen, tastiere, riconoscimento vocale) connesse direttamente con il sistema intelligente di controllo, basato su un'unità computerizzata centrale oppure basato su un sistema a intelligenza distribuita opportunamente interconnessa e organizzata con precise gerarchie.

Il sistema di controllo centralizzato, oppure l'insieme delle periferiche in un sistema a intelligenza distribuita, provvede a svolgere i comandi impartiti dall'utente (ad esempio l'accensione del calorifero o l'impostazione manuale/automatica della luce), a monitorare continuamente i parametri ambientali richiesti e a gestire in maniera autonoma alcune regolazioni (ad esempio la luce).

I sistemi di automazione sono di solito predisposti affinché ogni qualvolta venga azionato un comando, all'utente ne giunga comunicazione attraverso un segnale visivo di avviso/conferma dell'operazione effettuata (ad esempio LED colorati negli interruttori).

## Obiettivi del progetto

Le soluzioni tecnologiche che possono essere adottate per la realizzazione di un sistema domotico sono caratterizzate da peculiarità d'uso proprie degli oggetti casalinghi:

- \* Semplicità: il sistema domotico è diretto a un pubblico vasto e non professionale, per questo deve essere semplice da usare e avere interfacce molto intuitive; deve inoltre essere sicuro e non deve presentare pericoli per chi non ne conosce o comprende le potenzialità.

\* Continuità di funzionamento: il sistema deve essere costruito pensando al fatto che dovrà offrire un servizio continuativo ed essere semplice da riparare anche per personale non esperto o, nell'eventualità di un guasto, essere riparabile in breve tempo, fornendo nel frattempo almeno un servizio ridotto.

\* Basso costo: affinché un sistema domotico sia alla portata di tutti deve avere un costo contenuto, sia all'acquisto che in fase di manutenzione.

\* Automatizzazione di azioni quotidiane: il sistema deve semplificare alcune azioni quotidiane, soprattutto quelle ripetitive, non deve in alcun modo complicarle. Ciò costituisce un indispensabile ausilio per tutte quelle persone che sono soggette a limitazioni funzionali, anche temporanee (disabili, anziani, coloro che hanno subito un incidente ecc.)

Tutte queste caratteristiche, se non sviluppate singolarmente ma nel loro insieme, portano alla creazione di un sistema domotico integrato che può semplificare la vita all'interno delle abitazioni. La casa diventa intelligente non perché vi sono installati sistemi intelligenti, ma perché essi sono capaci di controllare e gestire in modo facile il funzionamento degli impianti presenti. Attualmente le apparecchiature tecnologiche sono poco integrate tra loro e il controllo è ancora ampiamente manuale; nella casa domotica gli apparati sono invece comandati da un unico sistema automatizzato che ne realizza un controllo intelligente.

# Capitolo 2: Descrizione del progetto

## L'idea

EH-1 nasce da un grande interesse personale per i sistemi automatici di controllo di ogni genere e per la fusione sempre più profonda tra l'uomo e il frutto delle sue ricerche tecnologiche. Al giorno d'oggi troviamo infatti una sempre maggior integrazione dell'elettronica nella nostra vita quotidiana, nel lavoro come nell'intimità della propria abitazione. Essendo sempre stato affascinato da questo fatto, ho scelto di affrontare personalmente la sfida di creare un sistema di gestione intelligente, moderno e aggiornabile, applicato a un piccolo modello di casa.

## La pianificazione

Al fine di sviluppare un progetto adeguato a un esame di Stato ho ritenuto sufficiente sviluppare un sistema di supporto all'utente in grado di controllare, rivelare e condizionare a piacimento l'ambiente abitativo, proteggendolo inoltre da minacce esterne. Tutto ciò prevede quindi un display per l'interfaccia uomo-macchina, vari sensori di luce e temperatura e alcuni utilizzatori in grado di intervenire in base alle impostazioni e ai parametri rilevati. È inoltre necessario un sistema di sicurezza di base che richieda l'inserimento di un codice tramite un tastierino numerico per avere accesso alle varie funzioni.

Le nozioni necessarie alla realizzazione di una casa elettronica non sono in gran parte contenute nel programma svolto nel corso di elettronica e telecomunicazioni nella scuola secondaria superiore, quindi ho dovuto prevedere un lungo periodo di tempo dedicato esclusivamente allo studio e all'utilizzo delle tecnologie impiegate.

Il livello di difficoltà del progetto non rende infatti possibile la realizzazione di un sistema che rispetti le caratteristiche elencate nel capitolo 1 con la normale logica integrata usata nei laboratori scolastici. Ho quindi scelto di utilizzare come unità di controllo i "PIC" (Programmable Interface Controller) prodotti dalla casa statunitense Microchip Technology, in particolare il modello 18F2685 (vedi più avanti la scelta componenti).

Per ridurre le dimensioni del sistema di gestione della casa, e per abbassare i costi di produzione per un'ipotetica produzione in serie, è previsto l'utilizzo (per quanto possibile) di componenti in formato SMD, ovvero Surface Mount Technology (tecnologia a montaggio superficiale).

## Le unità logiche di controllo

Si denomina "PIC" una famiglia di circuiti integrati a semiconduttore con tutte le funzioni di un microcontrollore. Ad oggi sono molto usati per il loro basso costo l'alto livello applicativo e la loro continua evoluzione tecnologica. Al loro interno troviamo diversi tipi di memoria integrata: memoria di programma, di Stack, RAM ed EEPROM, aggiornando l'obsoleta tecnologia CMOS in favore dell'odierna Flash. Sono inoltre presenti diversi moduli integrati di input/output paralleli e seriali, USART (Universal Synchronous-Asynchronous Receiver/Transmitter), oscillatore interno programmabile, convertitori ADC a 10 bit e canali PWM. Tramite appositi pin di programmazione è possibile scrivere sulla memoria di programma il listato di istruzioni esadecimali (comprensibili al microcontrollore) derivato da un programma scritto in linguaggio Assembly. Un grande vantaggio dei PIC consiste nell'esiguo numero di istruzioni necessarie alla sua programmazione (in media sono 33), ma ad oggi la gestione è semplificata ulteriormente dalla presenza sul mercato e in rete di software in vendita o gratuiti che consentono la scrittura di un programma in linguaggio C, convertendolo poi automaticamente in listato esadecimale.

I sistemi di controllo e automazione si possono fondamentalmente distinguere in due tipi; uno basato su un'unità di elaborazione centrale che permette di gestire tutte le attuazioni a partire dai risultati di rilevazione, e uno a struttura distribuita dove le interazioni avvengono localmente in maniera distribuita ed eventualmente comunicate a un'unità centrale per un controllo di coerenza generale. Ai fini del progetto ho scelto di suddividere le operazioni tra due PIC in costante contatto per l'implementazione delle funzioni più complesse. Uno prende il nome di CPU (Central Processing Unit), e si occupa della gestione del display, della misura della temperatura interna ed esterna della casa e del riscaldamento della stessa; svolge inoltre le funzioni di “master” nelle comunicazioni tra i due PIC. La seconda unità gestisce il sistema di sicurezza della casa, della rilevazione del livello luminoso dell'ambiente e della sua regolazione tramite emettitori luminosi a piacimento dell'utente; svolge inoltre le funzioni di “slave” nelle comunicazioni tra i due PIC.

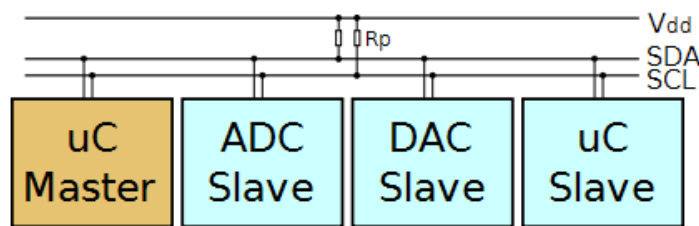
## Particolarità del progetto

### \* Protocollo di comunicazione seriale I<sup>2</sup>C

I<sup>2</sup>C (o IIC), acronimo di Inter Integrated Circuit, è un sistema di comunicazione seriale bifilare utilizzato tra circuiti integrati.

Il bus, sviluppato dalla Philips nel 1982, prevede in genere un dispositivo detto “master” che ha il controllo del clock di sistema e della comunicazione dati, al quale si sottopongono altri dispositivi secondari detti “slave”. Niente comunque vieta lo sviluppo di sistemi multi-master nei casi più complessi.

Il protocollo hardware dell'I<sup>2</sup>C richiede due linee seriali di comunicazione:



Logo standard I<sup>2</sup>C

\* SDA (Serial DATA line) per i dati

\* SCL (Serial Clock Line) per il clock (che rende sincrono il protocollo I<sup>2</sup>C)

Va aggiunta una connessione di riferimento detta impropriamente GND (non presente in figura) e una linea di alimentazione V<sub>DD</sub> a cui sono connessi i resistori di pull-up, che può anche non essere condivisa da tutti i dispositivi. Le tensioni tipiche usate sono +5 V o +3,3 V.

Come tutti i protocolli di comunicazione seriale il limite principale è rappresentato da una velocità di trasmissione molto inferiore a un protocollo parallelo. L'I<sup>2</sup>C è quindi usato per comunicare con dispositivi in cui semplicità e basso costo sono prioritari rispetto alla velocità di trasmissione.

Questo standard trova la sua applicazione progettuale nella comunicazione tra le due unità di controllo.

## \* Modulazione PWM

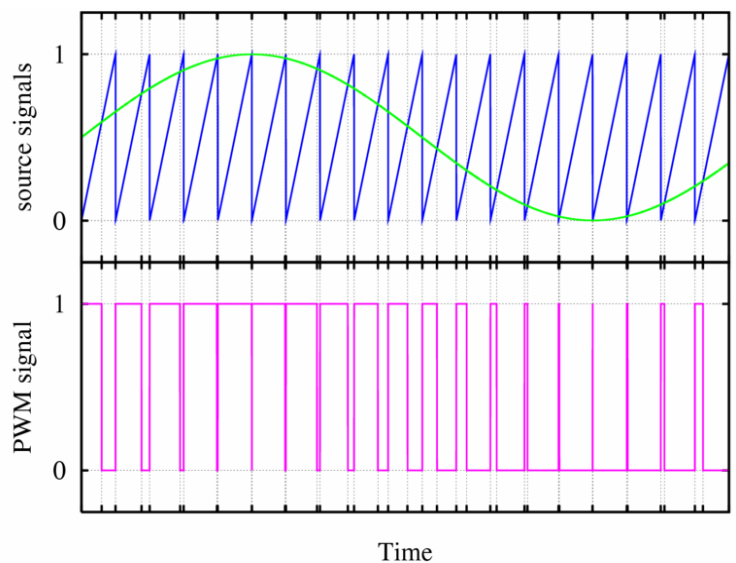
La modulazione di larghezza di impulso, detta PWM (Pulse-Width Modulation), è un tipo di modulazione digitale in cui l'informazione è codificata sotto forma di durata del periodo di ciascun impulso di un segnale. La durata di ciascun impulso può essere espressa in rapporto al periodo di clock, implicando il concetto di duty cycle. Un duty cycle pari a 0% indica un impulso di durata nulla, in pratica assenza di segnale, mentre un valore del 100% indica che l'impulso resta a livello alto per tutta la durata del clock.

La modulazione a larghezza di impulso è largamente utilizzata per regolare la potenza elettrica inviata a un carico, per esempio regolando la velocità dei motori in corrente continua e per variare la luminosità delle lampadine.

Come si può intuire, con un duty cycle pari a zero la potenza trasferita è nulla, mentre al 100% la potenza corrisponde al valore massimo trasferito nel caso non sia presente il circuito di modulazione. Ogni valore intermedio determina una corrispondente fornitura di potenza.

Il vantaggio di questa tecnica è di ridurre drasticamente la potenza dissipata dal circuito limitatore rispetto all'impiego di transistor controllati analogicamente. In un semiconduttore la potenza dissipata è determinata dalla corrente che lo attraversa moltiplicata per la differenza di potenziale presente ai suoi capi. In un circuito PWM il transistor in un istante conduce completamente, riducendo al minimo la caduta ai suoi capi, oppure non

conduce, annullando la corrente, ed in entrambi i casi la potenza dissipata è minima.



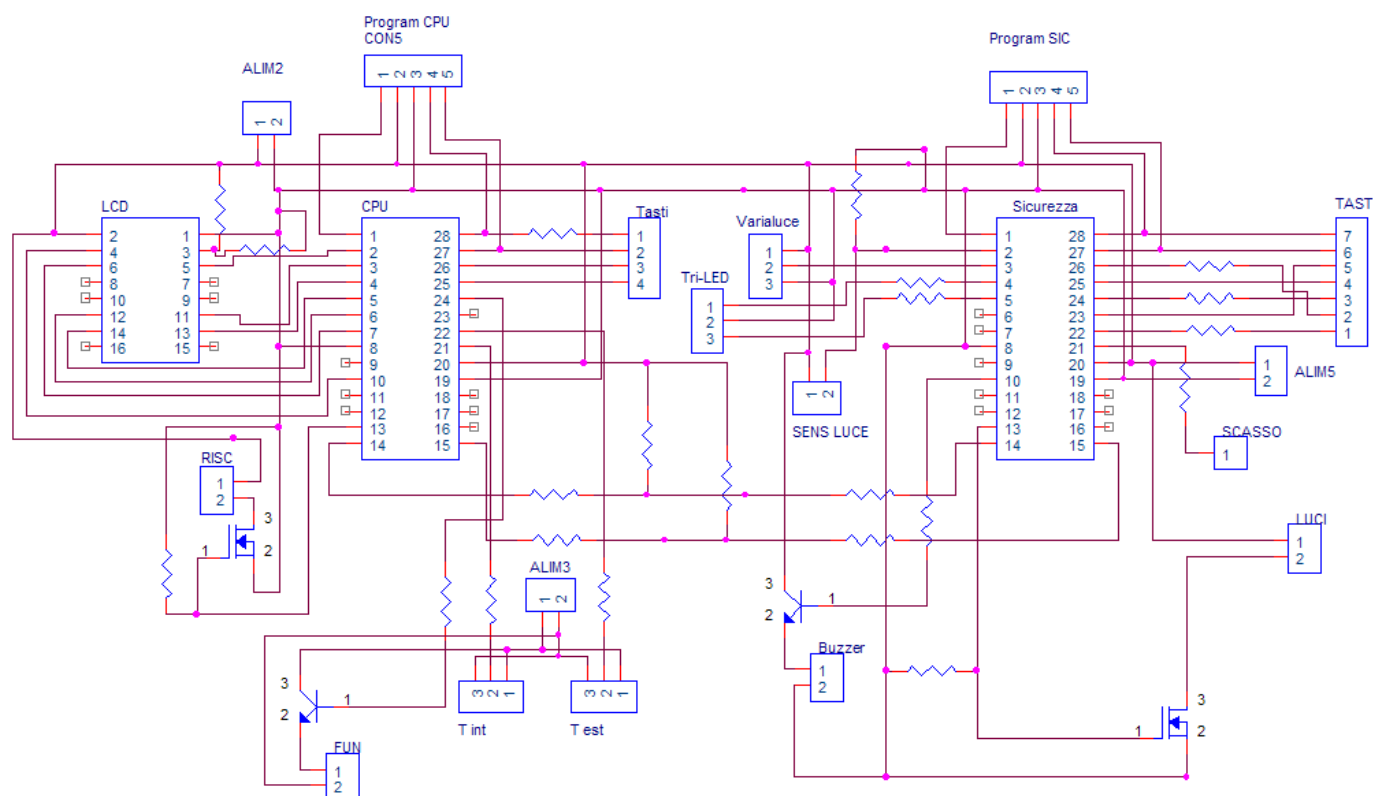
In figura vediamo un segnale sinusoidale modulato tramite larghezza di impulso in funzione di una sinusoide.

La modulazione PWM verrà utilizzata in questo progetto per la gestione dell'illuminazione e del riscaldamento. Si può in questo modo simulare un segnale analogico che alimenta i due utilizzatori variando semplicemente il duty cycle di un segnale digitale, risparmiando l'uso di un DAC con tutti i relativi problemi. Si precisa però che la corrente in uscita da un PIC è nell'ordine dei 20mA, quindi per l'illuminazione e il sistema di riscaldamento serve un sistema di amplificazione.

# Capitolo 3: Realizzazione del progetto

## Schema elettrico

Una volta stilato un progetto di massima della casa intelligente, ho realizzato lo schema elettrico del sistema usando il programma di disegno e simulazione OrCad Express, appartenente alla suite di disegno assistito OrCad 9.0. Il disegno evidenzia la distinzione dei due blocchi operativi e l'esiguo numero di componenti impiegati (esclusi connettori e le resistenze). Ciò si deve al fatto che la gestione dell'intero sistema è affidata ai due PIC, programmati via software (in linguaggio C). Sopra ogni PIC troviamo un connettore per la programmazione dello stesso, mentre agli estremi del disegno vediamo il display LCD (a sinistra) e il tastierino numerico.



Title		
Schema elettrico EH-1		
Size	Document Number	Rev
A	Marco Di Goro VB ELT 2007/08	
Date:	Sunday, June 08, 2008	Sheet 1 of 1

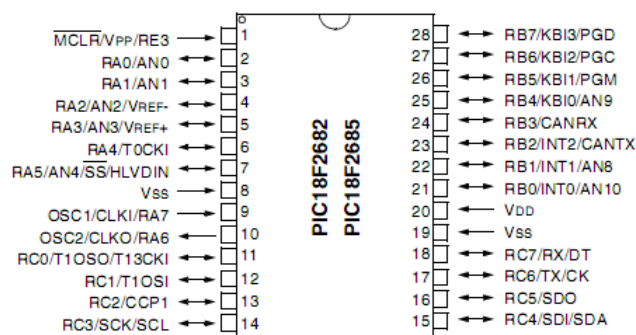
## Scelta dei componenti

Per realizzare un buon progetto è necessario operare una giusta scelta dei componenti, trovando un compromesso che riduca il più possibile i costi di produzione e che consenta di ottenere le prestazioni migliori. Ecco quindi un elenco dei componenti impiegati con motivazione e difficoltà di scelta.

### \* PIC 18F2685

Tra le molte centinaia di PIC di casa Microchip ho scelto questo per le sue eccezionali caratteristiche in fatto di prestazioni, costo, durata e praticità di utilizzo del package SMD. Ecco un riassunto delle sue principali caratteristiche.

- Oscillatore interno da 31 KHz a 8 MHz
- Fino a 1.000.000 di riscrittura sulla EEPROM
- Moltiplicatore hardware interno
- 10 canali ADC (via multiplexer) a 10 bit
- Modulo CCP (Capture/Compare/PWM)
- Modulo I<sup>2</sup>C master/slave
- Modulo ECAN (Enhanced Controller Area Network)
- NanoWatt Technology
- 96 Kbytes di Memoria di programma



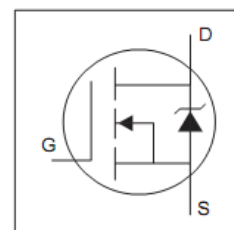
Durante tutta la fase di progettazione questo componente non ha mai dato problemi di alcun genere; si è anzi dimostrato al di sopra delle aspettative in più occasioni per la sua affidabilità, praticità e potenza.



### \* IRLR 2705

Ho scelto questo MOS-FET perché ha ottime qualità di conduzione in saturazione. In questa condizione infatti la sua resistenza interna è nell'ordine di pochi milli-ohm, il che comporta una ridottissima dissipazione di potenza. Un'altra caratteristica fondamentale di questo componente è che il gate segue la logica TTL. Questo lo porta a operare solo da interruttore ON/OFF senza bisogno di un driver a BJT, e unitamente alla sua alta velocità di switching, lo rende adatto alle applicazioni con segnali impulsivi.

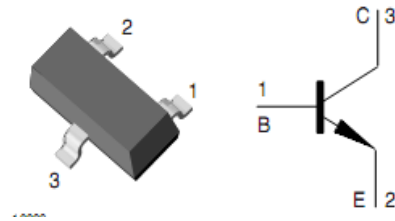
Dovendo controllare in PWM il sistema di illuminazione e di riscaldamento, il MOSFET IRLR 2705 si presta perfettamente a funzionare da amplificatore del segnale digitale in ingresso al Gate. Il costruttore inoltre raccomanda di porre una resistenza di pull-down sul Gate.





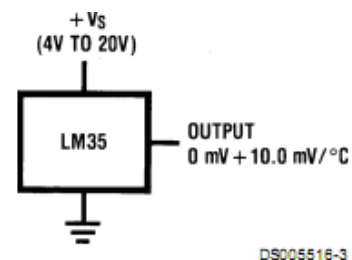
### \* Transistor NPN BC-817

Per la gestione del buzzer a 5V (non in PWM) ho scelto questo transistor perché ha un costo molto contenuto, e nonostante le ridottissime dimensioni del suo package SMD (SOT-23) riesce a sopportare correnti di collettore fino a 1 A.



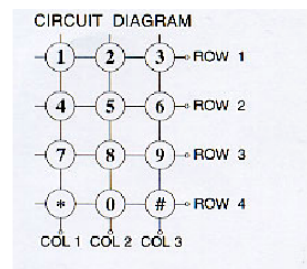
### \* Sensore di temperatura LM35DZ

Ho scelto questo sensore di temperatura per via del costo ridotto, del fatto che è calibrato in gradi Celsius e non necessita di alcuna conversione (al contrario di altri). Ha una buona precisione, assorbe solo qualche decina di micro-ampere e opera tra 4 e 30 Volt; inoltre fornisce in uscita un segnale analogico lineare secondo la legge  $V_o = 0 + 10 \text{ mV}/^\circ\text{C}$ .  
Range: 2-150°C



### \* Tastierino numerico 3x4

Si basa su un intreccio di righe e colonne come in figura. A ogni tasto premuto corrisponde una diversa combinazione di una colonna e una riga cortocircuitate. Tramite un'adeguata routine di gestione è quindi possibile stabilire quale sia il tasto premuto. Garantisce inoltre una durata molto lunga.



AK-207

OUTPUT ARRANGEMENT	
OUTPUT PIN NO.	SYMBOL
1	COL 2
2	ROW 1
3	COL 1
4	ROW 4
5	COL 3
6	ROW 3
7	ROW 2

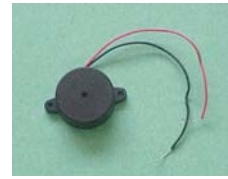
### \* Resistenze SMD

Viste le ridottissime dimensioni del package 1206 scelto per queste resistenze (comunque possibili da saldare a mano) e il basso costo, sono ovviamente migliori delle resistenze tradizionali. Sono inoltre caratterizzate da una tolleranza molto più ristretta.



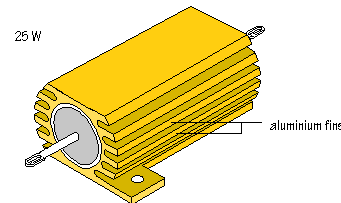
### \* Buzzer 5V DC

È un semplice buzzer a 5V di basso costo.



### \* Resistenza di potenza

Per il sistema di riscaldamento ho scelto una resistenza di potenza (10Ω – 25W) con buone capacità di dissipazione grazie al suo case in lega termo-conduttiva.



### \* Fotoresistenza NORPS-12

Per l'acquisizione del livello di luce ambientale ho scelto questa fotoresistenza. Ecco un estratto del datasheet contenete le principali caratteristiche.

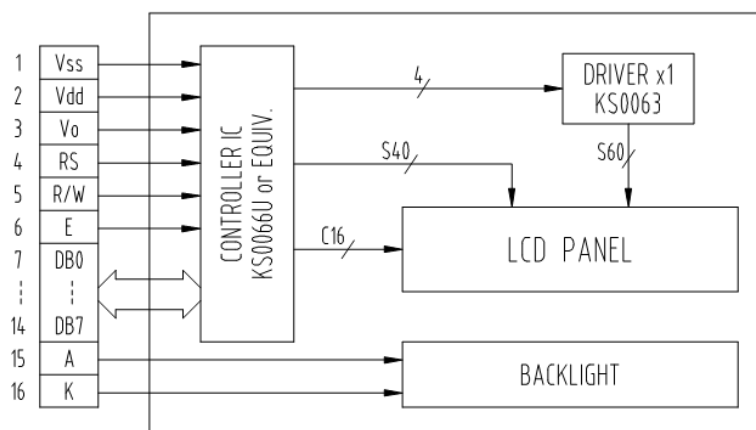


**Electrical Characteristics** ( $T_A=25^\circ\text{C}$  unless otherwise noted)

Symbol	Parameter	Min.	Typ.	Max.	Units	Test Conditions
$R_L$	Light Resistance	5.4		12.6	$k\Omega$	1 ftc. (1)
$R_D$	Dark Resistance	1.0			$M\Omega$	15 sec. after removal of test light.
$\lambda_P$	Spectral Peak		550		nm	

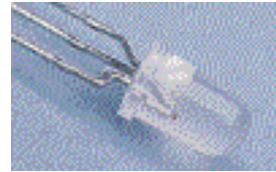
### \* Display LCD 20x2 standard Hitachi con logica integrata

Grazie alla sua logica integrata un display LCD a standard Hitachi consente un interfacciamento completo con soli 16 piedini, di cui 4 di alimentazione, 1 per la regolazione del contrasto, 3 di controllo e 8 di dati. Nel datasheet del componente sono elencate tutte le procedure e le combinazioni necessarie a un corretto funzionamento del display, tuttavia in fase di programmazione si utilizzerà un'apposita routine di gestione che ne semplifica estremamente l'utilizzo.



### \* Led tricolore verde-rosso-arancione

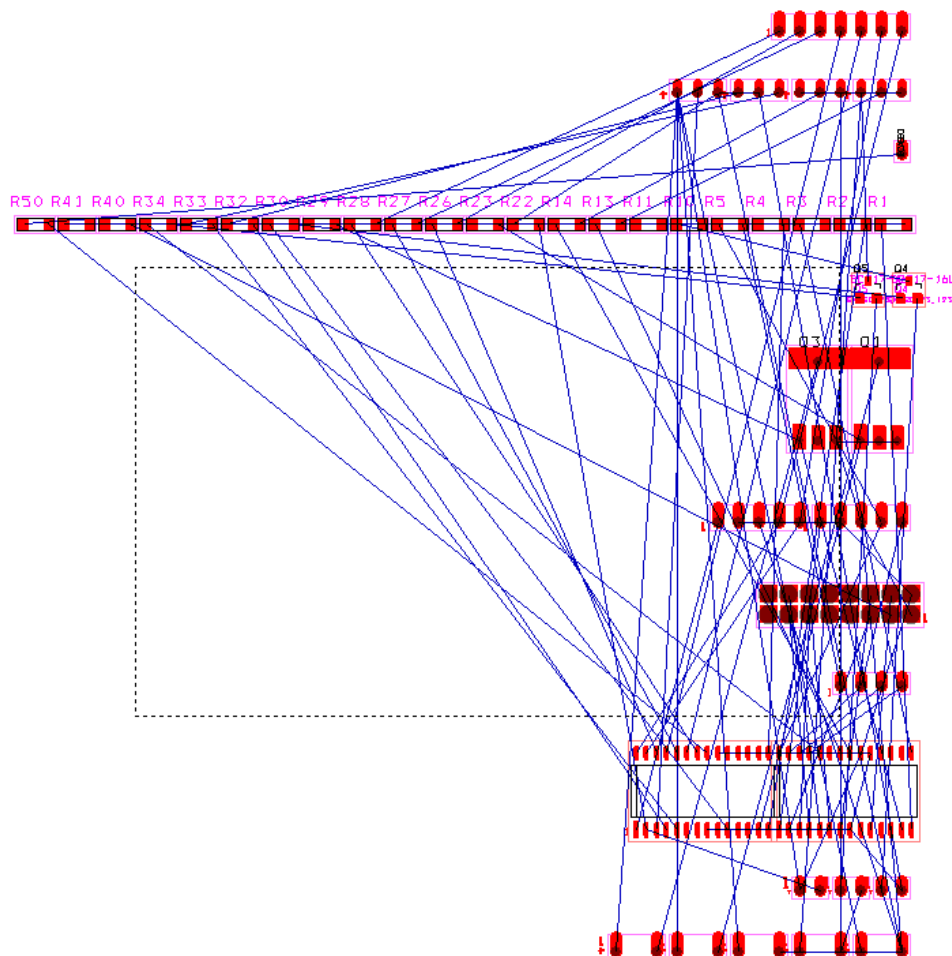
Per supportare la gestione del tastierino numerico ho scelto di usare questo led tricolore a catodo comune centrale. Alimentando uno dei due pin laterali si ottiene un colore rosso o verde. Alimentandoli contemporaneamente i due colori si sommano, ottenendo così un colore arancione.



## Circuito stampato

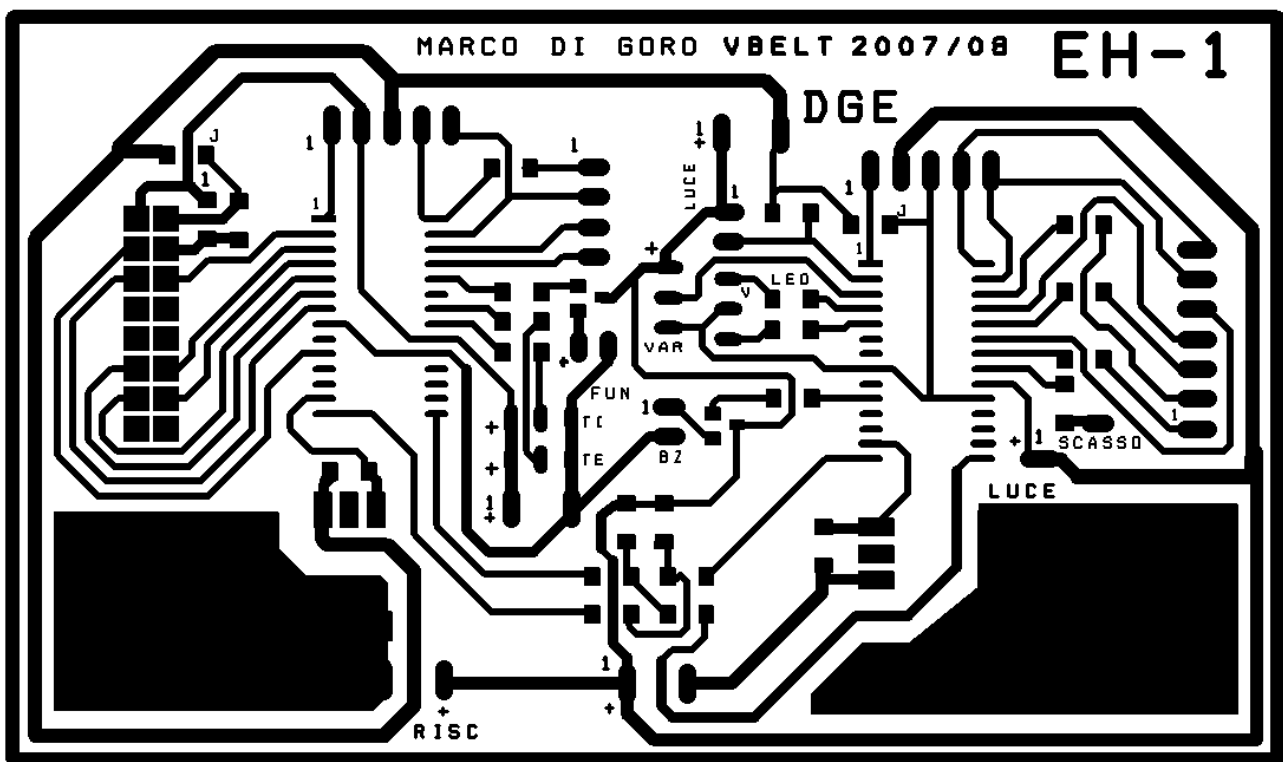
Dopo la realizzazione dello schema elettrico del progetto e la scelta dei componenti adeguati, si associa ad ognuno di essi un “footprint” (impronta), ovvero la forma di un’adeguata piazzola di rame che ne permetta la saldatura alla scheda. Inserendo quindi questi footprint in un’apposita libreria e, associandoli ai rispettivi componenti, si procede alla creazione di una “netlist” che ne elenchi tutte le connessioni. Il passo successivo consiste nel creare un nuovo disegno importando la netlist con il software OrCad Layout (sempre appartenente al pacchetto OrCad 9.0); a questo punto i vari footprint dei componenti scelti vengono disposti nel disegno, unendoli però tra di loro secondo le connessioni stabilite nello schema elettrico (linee blu).

Ecco quello che vediamo una volta importata la netlist del progetto:



Resta ancora molto lavoro da fare. Occorre scegliere un'adeguata disposizione dei componenti e collegarli disegnando le piste di rame che porteranno il collegamento elettrico. Qui entrano in gioco le linee blu in figura, che danno un'idea dei collegamenti da fare. È necessario trovare un compromesso tra la realizzazione di una scheda di dimensioni ridotte e il disegno di piste possibilmente brevi, dirette e disposte su un numero ridotto di "layer" (piani). Si deve anche tenere conto del flusso di corrente che attraversa ogni pista, che dovrà essere quindi dimensionato di conseguenza. Nello specifico, a parte le piste di alimentazione, le correnti in gioco sono di pochi milli-ampere, e quindi si possono tranquillamente realizzare piste di 0,7mm di larghezza.

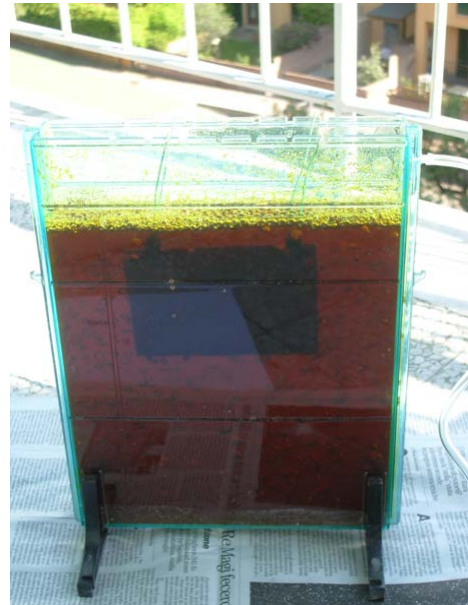
Si procede quindi come detto in precedenza a "sbrogliare" il circuito, ossia a effettuare tutti i collegamenti necessari senza incrociare mai due piste che non debbano essere collegate tra loro da progetto, e questo tramite numerose prove al fine di scegliere la soluzione ottimale. Dopo numerosi tentativi si ottiene lo sbrogliato riportato di seguito. Utilizzando occasionalmente delle resistenze da 0Ω come ponticelli, è possibile completare il lavoro su un solo "layer" (piano), abbassando notevolmente i costi di produzione rispetto ad uno sbrogliato su doppia faccia. Ho previsto anche due ampie zone di dissipazione in corrispondenza dei due mosfet per meglio smaltire il calore sviluppato per effetto Joule.



Il passo successivo consiste nello stampare a laser i due disegni su fogli acetati (plastica trasparente) e procedere con la tecnica della “fotoincisione”. Il processo consiste nel posizionare adeguatamente i disegni sopra a una scheda di rame ricoperto da un composto organico detto “fotoresist”, esponendo poi il tutto ai raggi UV tramite il “bromografo” (un sistema di lampade UV gestito elettronicamente). La scheda viene poi immersa in una vasca di soda caustica ( $\text{NaOH}$ ) che asporta solo il fotoresist impressionato (poiché non coperto dal disegno), e successivamente in una vasca contenente tricloruro ferrico ( $\text{FeCl}_3$ ), un acido che corrode il rame non protetto dal fotoresist. Alla fine di questo processo si lava la scheda, si toglie il fotoresist rimasto con alcool e si ricopre di stagno il disegno di rame.

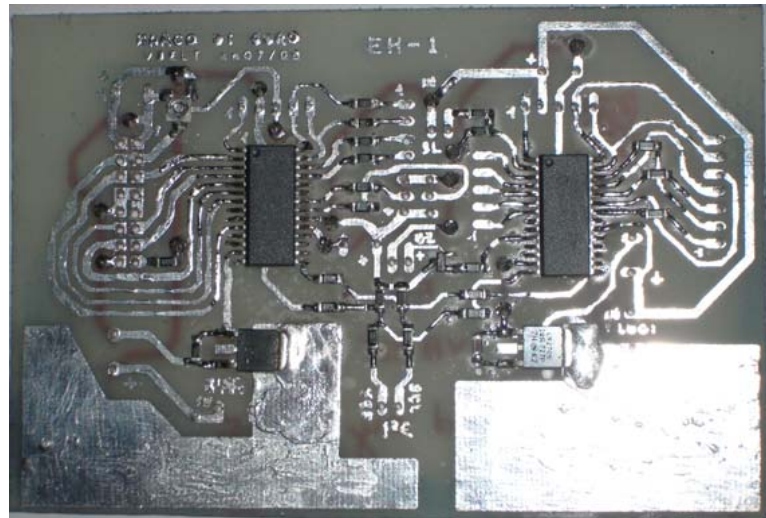
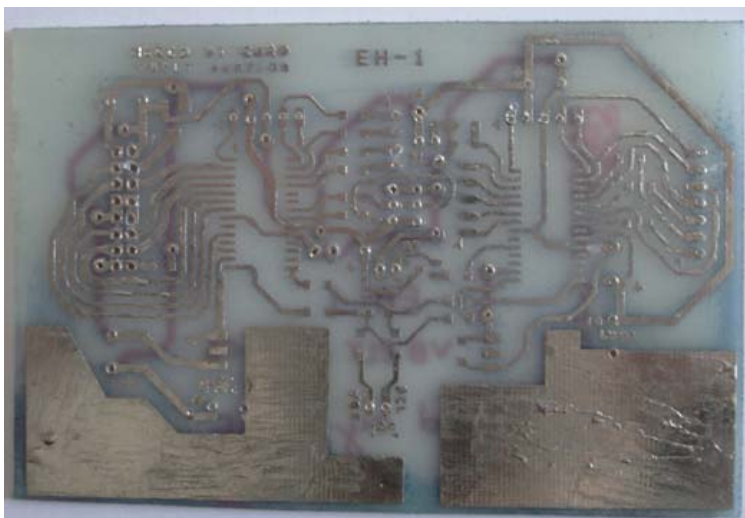


Bromografo



Immersione nell'acido

A questo punto si procede con la foratura della scheda, la saldatura dei vari componenti e connettori, e con i collegamenti tra le due facce della scheda. Ho preferito lasciare per ultimi i PIC, vista la loro sensibilità alle alte temperature raggiunte nella fase di saldatura.





# Capitolo 4: Il software

## Programmi utilizzati

### \* CCS PICC 4.057

È un compilatore in linguaggio C per i PIC di casa Microchip. Ho scelto di utilizzare questo programma perché ha prestazioni eccezionali e supporta tutti i microcontrollori delle serie 12, 16 e 18, sia con tecnologia CMOS che Flash. Come detto in precedenza ho dovuto studiare autonomamente la programmazione in C dei PIC, ma l'ottima funzione di help inserita nel programma mi ha aiutato notevolmente ad acquisire la conoscenza necessaria delle istruzioni per realizzare il software del progetto. Il programma consente il cosiddetto ICD (In-Circuit Debug) con soli due collegamenti, e contiene un gran numero di routines di gestione già scritte (come quella per il controllo del display LCD utilizzata nel progetto). Il punto di forza di questo programma è sicuramente l'ottima procedura guidata per la creazione di un nuovo progetto, che risparmia moltissimo lavoro all'utente.

### \* Microchip MPLab 7.0

MPLAB Integrated Development Environment (IDE) è un programma gratuito adibito alla programmazione dei PIC e al debug del software. Si interfaccia con molti programmatori, anche di altre case costruttrici, e supporta la programmazione di tutti i microcontrollori della Microchip. È inoltre capace di importare un gran numero di formati di codice sorgente. In questo progetto ho utilizzato questo programma per interfacciare il PC ai PIC tramite il programmatore descritto di seguito.

## Programmatore Microchip “MPLab ICD2”

MPLAB ICD 2 è un programmatore in-circuit Microchip per dispositivi flash che consente anche il debugging del programma. Il programma realizzato può essere eseguito in tempo reale (visualizzandolo su MPLab), esaminato in dettaglio e debuggato.

Caratteristiche principali:

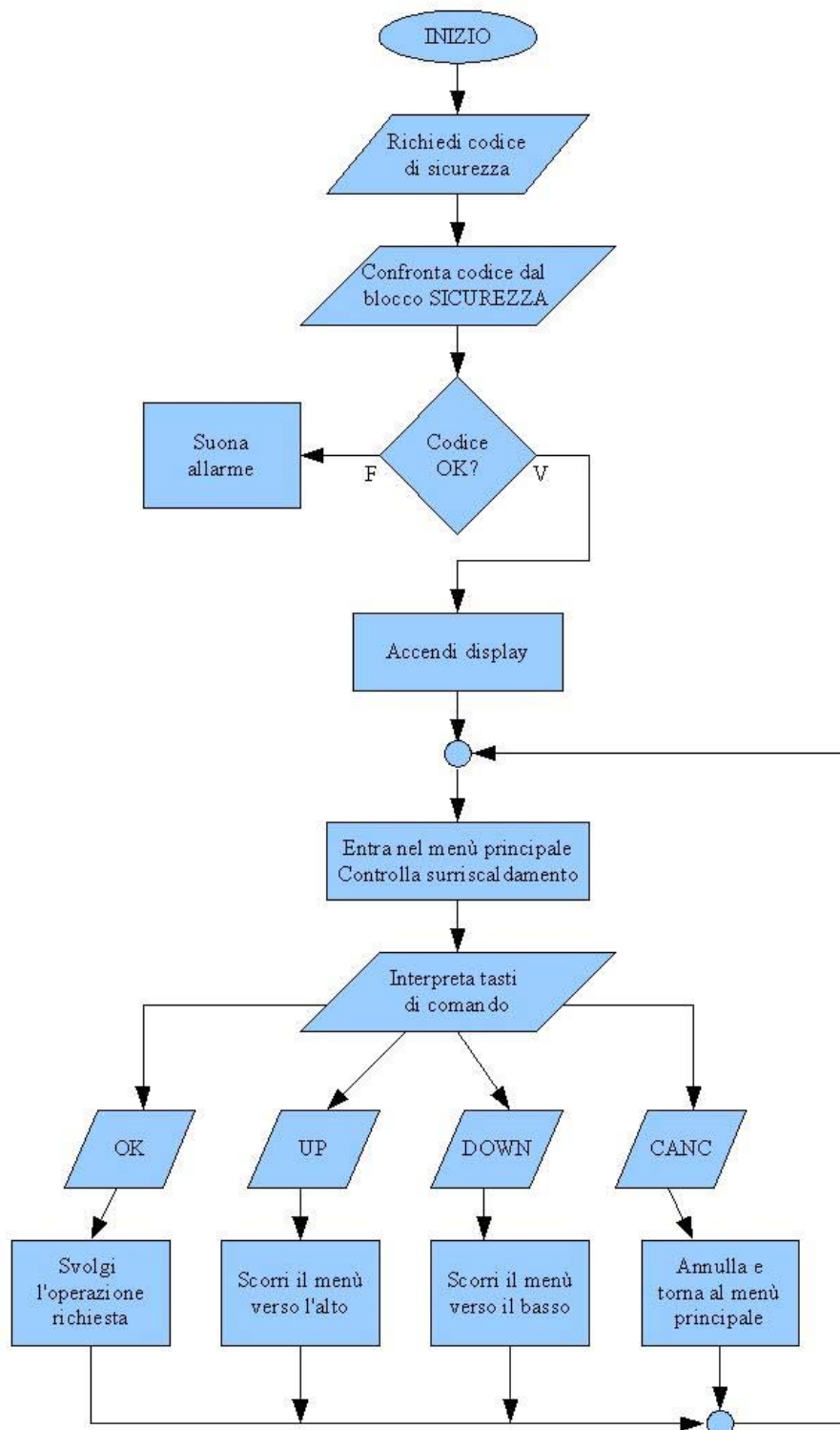
- Interfaccia USB (2 Mbits/s) o RS232;
- Monitoraggio delle tensioni di lavoro;
- Firmware aggiornabile;
- Hardware particolarmente compatto;
- Tensioni supportate: da 2 a 6 V;
- Led di diagnostica (Power, Busy, Error);
- Compatibile con le famiglie PIC10/12F, PIC16F, PIC18F e dSPIC30F.



# Codice sorgente unità CPU

## \* Diagramma di flusso

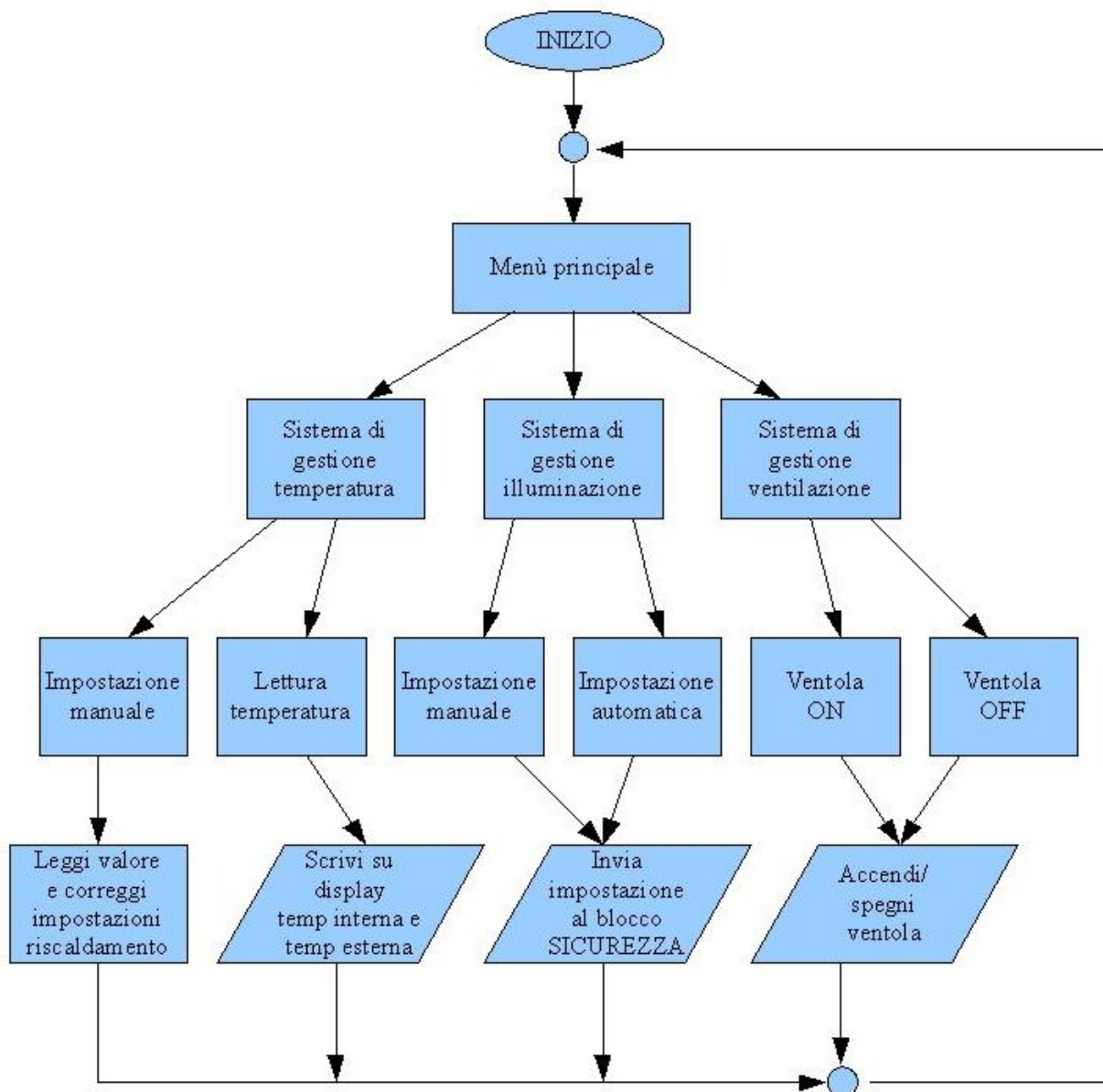
Per ragioni di spazio ho inserito solo il diagramma di flusso. Il codice sorgente è allegato in fondo al documento.



### \* Diagramma di flusso delle funzioni integrate per il display LCD

Dal menù principale di gestione della casa si diramano altri menù secondari che danno accesso a funzioni diverse:

- regolazione della temperatura interna della casa
- visualizzazione della temperatura interna ed esterna
- impostazione della regolazione di illuminazione (automatica/manuale)
- accensione/spegnimento ventola di raffreddamento



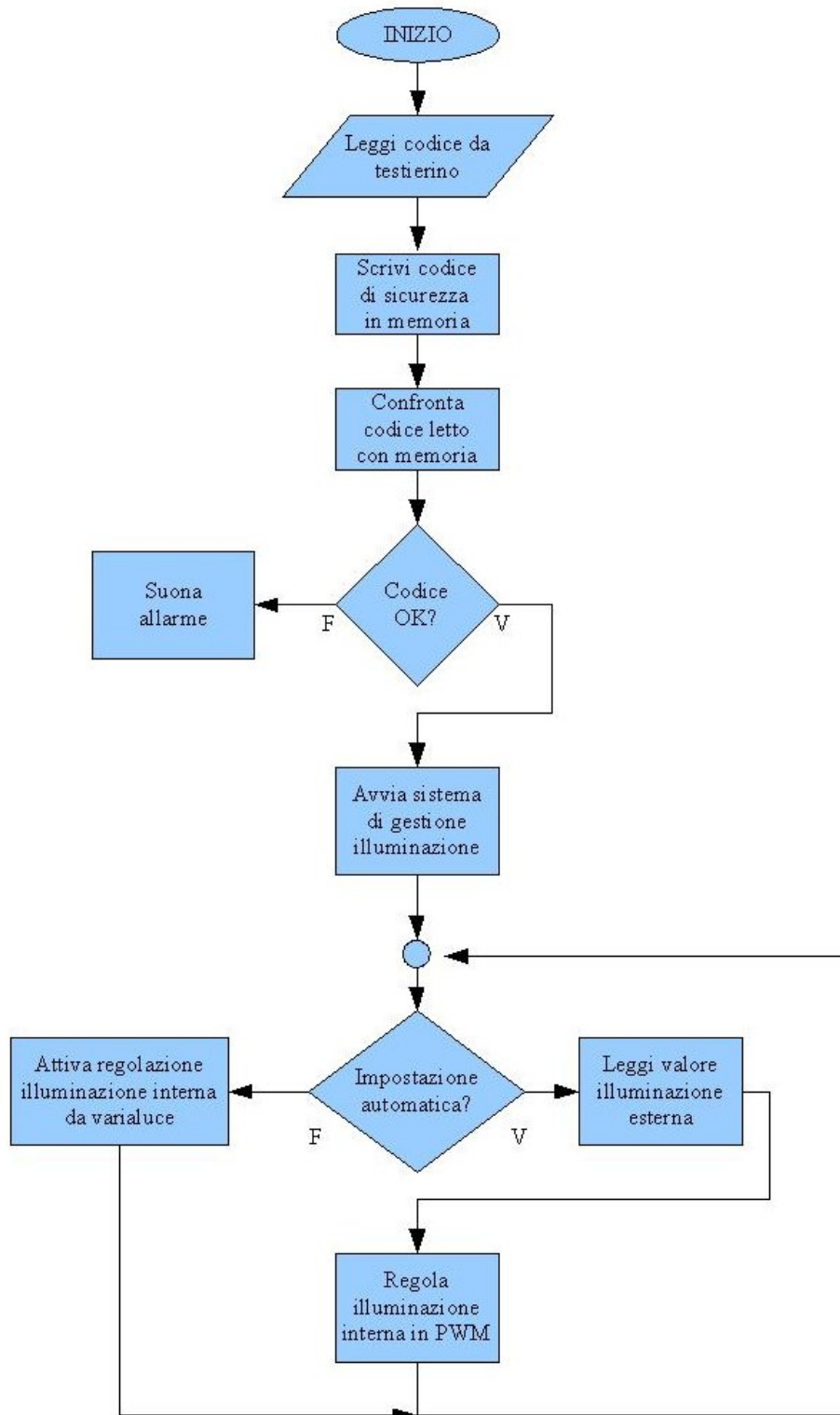
Come si vede dallo schema c'è un'abbondante quantità di sottomenù da gestire da parte del PIC CPU e, considerato che i quattro tasti di comando (OK, UP, DOWN, CANC) permettono quattro tipi diversi di "passaggio" da una funzione all'altra, il programma di gestione del display LCD si rivela piuttosto complesso, come si vedrà più avanti.



# Codice sorgente unità Sicurezza

## \* Diagramma di flusso

Come per l'unità CPU anche in questo caso, per ragioni di spazio, ho inserito solo il diagramma di flusso. Il codice sorgente è allegato in fondo al documento.



## “Problem Solving”

Come era prevedibile il funzionamento del progetto non è stato immediato. Sono state necessarie molte correzioni per giungere a una piena operatività. Ho incontrato diversi problemi nella gestione del tastierino numerico perché la routine di gestione integrata nel compilatore PICC 4.057 purtroppo si è rivelata incompleta e non funzionante. Ho quindi dovuto realizzarne una completamente nuova in sostituzione dell'originale, e questo ha richiesto circa una settimana, in quanto risultava difficile la gestione in contemporanea di pull-up e interrupt nella routine in questione. Ho dovuto inoltre prevedere delle resistenze aggiuntive nelle piste di connessione diretta tra PIC. Altre difficoltà sono derivate dallo studio e utilizzo del protocollo I<sup>2</sup>C, rivelatosi più complicato del previsto. Anche la lavorazione delle superfici della casa per l'installazione dei dispositivi e del cablaggio hanno richiesto molte ore di lavoro, vista la loro complessità (vedi foto). La fase più impegnativa della risoluzione dei problemi (in termini di lavoro e tempo) è stata il “debug” dell'intero codice sorgente che avevo scritto, soprattutto in fase di comunicazione PIC-PIC e PIC-periferiche.



# Capitolo 5: Considerazioni finali

La realizzazione del progetto ha richiesto diversi mesi, sebbene avessi programmato tempi più brevi. Una delle cause dei continui ritardi è stata l'iniziale mancanza delle nozioni necessarie alla progettazione di un sistema demotico interattivo, poiché non previste nel corso di elettronica e telecomunicazioni della scuola secondaria superiore. Ho impiegato molto tempo nella sperimentazione di semplici programmi al fine di imparare a interfacciare un PIC con periferiche esterne, con altri PIC, e persino con se stesso. Posso senz'altro affermare che la "progettazione da zero" è una sfida enorme, così come lo è stata questa esperienza, sviluppata senza aiuti esterni dall'ideazione al prodotto finito. A parte la realizzazione del progetto in sé, ho avuto la possibilità di ampliare considerevolmente le mie conoscenze nella programmazione e nell'interfacciamento dei microcontrollori, che rappresentano uno dei maggiori settori di sviluppo dell'elettronica di oggi e di domani. Ho anche imparato, dopo numerosi tentativi, a realizzare con facilità dei circuiti stampati partendo dallo schema elettrico, a effettuare saldature su componenti in formato SMD (che richiede precisione e abilità superiori al tradizionale formato P-DIP) e a fare una stima e una scelta dei componenti necessari. Detto questo, considerato anche che la casa alla fine è divenuta "intelligente", posso dire anche di essermi divertito.

*Marco Di Goro*

## Capitolo 6: Appendice al capitolo 4

Per semplicità ho scelto di omettere le routines di gestione del display LCD e del tastierino numerico, in quanto non sono determinanti ai fini del progetto. In alcuni casi è stato necessario spezzare in due alcune righe del codice. Per distinguerle ho aggiunto il simbolo \*\*\* in fondo a ciascuna di esse.

## Codice sorgente unità CPU scritto con CCS PICC 4.057

```
#include <CPU1.h>
#include <LCD_CPU.c>           //Funzione che permette la
                                //gestione del display

//Global Defines
#define UP    PIN_B4           //Scorrere opzioni del menù
#define DOWN  PIN_B5
#define OK    PIN_B6           //Selezione opzioni
#define CANC  PIN_B7           //Torna al livello superiore del menù
#define VENTO PIN_B3           //PIN che attiva la ventola

//Global Variables
int codice, funzione, temp, M_A, contr;
float temp_int, temp_est;

//Function Prototypes

void controllo();
void standby();               //Function n°1
void temperature();           //Function n°2
void light();                  //Function n°3
void fun();                    //Function n°4
void manual_set();             //Function n°5
void auto_set();               //Function n°6
void show();                   //Function n°7
void temp_select();            //Function n°8
void imp_temp();               //Function n°11
void imp_fun();                //Function n°12

void main()
{
    port_b_pullups(TRUE);
    //Connettere i quattro tasti a massa
    setup_adc_ports(ALL_ANALOG|VSS_VDD);
    setup_adc(ADC_CLOCK_INTERNAL|ADC_TAD_MUL_2);
    setup_wdt(WDT_OFF);
    setup_timer_0(RTCC_INTERNAL);
    setup_timer_1(T1_DISABLED);
    setup_timer_2(T2_DIV_BY_4,249,1);
    setup_timer_3(T3_DISABLED|T3_DIV_BY_1);
    setup_ccp1(CCP_PWM);
    setup_vref(FALSE);

    setup_oscillator(OSC_4MHZ|OSC_INTRC|OSC_31250|OSC_PLL_OFF);

    lcd_init();
    delay_ms(250);
    lcd_putc("\f");
    lcd_putc(" Benvenuto ");
    delay_ms(1500);
    lcd_putc("\f");
    lcd_putc("Inserire il codice e");
    lcd_putc("\n");
    lcd_putc(" poi premere OK ");
    while(input_state(OK)==1)    //{
        {delay_ms(10);}
    }

    i2c_start();
    i2c_write(0xA1);           // Set Slave address & request data
    codice=i2c_read();         // Now read from slave.
    i2c_stop();                //If the code is correct the function will returns 1
    lcd_putc("\f");
    delay_ms(500);
    if(codice==1)
    {
        lcd_putc(" Codice corretto ");
        lcd_putc("\n");
        lcd_putc(" Bentornato a casa ");
        delay_ms(1000);
        lcd_putc("\f");
    }
    else
    {
        while(1)
        {
            lcd_putc(" Codice errato ");
            delay_ms(750);
            lcd_putc("\n");
            lcd_putc(" !!!ALLARME!!! "); //Scritta lampeggiante
            delay_ms(750);
            lcd_putc("\f");
        }
    }
    funzione=1;
    while (1)
    {
        controllo();
        switch (funzione)
        {
            case 1: standby();
                    break;
            case 2: temperature();
                    break;
            case 3: light();
                    break;
            case 4: fun();
                    break;
            case 5: manual_set();
                    break;
            case 6: auto_set();
                    break;
            case 7: show();
                    break;
            case 8: temp_select();
                    break;
            case 11: imp_temp();
                    break;
            case 12: imp_fun();
                    break;
            default: standby();
                    break;
        }
    }
}
```

```

}

void controllo()
{
    set_adc_channel(10);
    temp_int=read_adc();
    temp_int=temp_int/1023*5;
    //Conversione dato binario a 5V
    temp_int=(temp_int*10/1000)+2;
    //converte in °C e aggiunge 2°C
    delay_ms(10); //perché il range del sensore parte da 2°C
    set_adc_channel(8);
    temp_est=temp_est/1023*5;
    //Conversione dato binario a 5V
    temp_est=(temp_est*10/1000)+2;
    //converte in °C e aggiunge 2°C
    delay_ms(10); //perché il range del sensore parte da 2°C
    switch(temp)
    {
        case 35: if(temp_int<temp)
            {
                set_pwm1_duty(307);
            }
            else
            {
                set_pwm1_duty(0);
            }
            break;
        case 45: if(temp_int<temp)
            {
                set_pwm1_duty(818);
            }
            else
            {
                set_pwm1_duty(0);
            }
            break;
        default: set_pwm1_duty(0);
            break;
    }
}

void standby()
{
    lcd_putc("\f");
    lcd_putc("EH-1 Management Menù");
    delay_ms(750);
    lcd_putc("\n");
    lcd_putc(" by DG Engineering ");
    contr=0;

    while(input_state(UP)==1&&input_state(DOWN)==1 ***
        &&input_state(OK)==1)
    {
        delay_ms(10);
        contr=contr+1;
        //Ogni 5 secondi viene comunque effettuato
        if(contr<=500) //il controllo del riscaldamento per
            {controllo();} //prevenire surriscaldamenti indesiderati
    }
    funzione=2;
}

void temperature()
{
    lcd_putc("\f");
    lcd_putc("Gestione temperatura");
    while(input_state(UP)==1&&input_state(DOWN)==1&& ***
        input_state(OK)==1&&input_state(CANC)==1)
        //Aspetta pressione di un tasto

```

```

    {}

    if(input_state(OK)==0)
    {funzione=11;} //Impostazione manuale
    if(input_state(UP)==0)
    {funzione=4;} //Ventilazione
    if(input_state(DOWN)==0)
    {funzione=3;} //Luce
    if(input_state(CANC)==0)
    {funzione=1;} //Ritorno allo standby
    }

    void light()
    {
        lcd_putc("\f");
        lcd_putc(" Gestione impianto ");
        lcd_putc("\n");
        lcd_putc(" di illuminazione ");
        while(input_state(UP)==1&&input_state(DOWN)==1&& ***
            input_state(OK)==1&&input_state(CANC)==1)
        {
        }
        if(input_state(OK)==0)
        {funzione=5;} //Impostazione manuale
        if(input_state(UP)==0)
        {funzione=2;} //Gestione temperatura
        if(input_state(DOWN)==0)
        {funzione=4;} //Ventilazione
        if(input_state(CANC)==0)
        {funzione=1;} //Ritorno allo standby
    }

    void fun()
    {
        lcd_putc("\f");
        lcd_putc(" Gestione impianto ");
        lcd_putc("\n");
        lcd_putc(" di ventilazione ");
        while(input_state(UP)==1&&input_state(DOWN)==1&& ***
            input_state(OK)==1&&input_state(CANC)==1)
        {
        }
        if(input_state(OK)==0)
        {funzione=12;} //Ventola ON/OFF
        if(input_state(UP)==0)
        {funzione=3;} //Gestione illuminazione
        if(input_state(DOWN)==0)
        {funzione=2;} //Gestione temperatura
        if(input_state(CANC)==0)
        {funzione=1;} //Ritorno allo standby
    }

    void imp_temp() //Impostazione temperatura
    {
        lcd_putc("\f");
        lcd_putc("Impostazione manuale ");
        lcd_putc("\n");
        lcd_putc(" della temperatura ");
        while(input_state(UP)==1&&input_state(DOWN)==1&& ***
            input_state(OK)==1&&input_state(CANC)==1)
        {
        }
        if(input_state(OK)==0)
        {funzione=8;} //Imposta temperatura
        if(input_state(UP)==0)
        {funzione=7;} //Visualizza temperatura
        if(input_state(DOWN)==0)
        {funzione=7;}
        if(input_state(CANC)==0)
        {funzione=1;} //Ritorno allo standby
    }

```

```

void manual_set()
{
    lcd_putc("\f");
    lcd_putc("Regolazione manuale ");
    lcd_putc("\n");
    lcd_putc(" dell'illuminazione ");
    while(input_state(UP)==1&&input_state(DOWN)==1&& ***
        input_state(OK)==1&&input_state(CANC)==1)
    {}
    if(input_state(OK)==0)
    {
        M_A=0;
        i2c_start();    //Regolazione manuale
        i2c_write(0xA0); // Set Slave address & send data
        i2c_wite(M_A);   // Now send from slave.
        i2c_stop();
        funzione=1;
    }
    if(input_state(UP)==0)
    {funzione=6;}      //Impostazione automatica
    if(input_state(DOWN)==0)
    {funzione=6;}      //Impostazione automatica
    if(input_state(CANC)==0)
    {funzione=1;}      //Ritorno allo standby
}

```

```

void auto_set()
{
    lcd_putc("\f");
    lcd_putc("Regolaz. automatica ");
    lcd_putc("\n");
    lcd_putc(" dell'illuminazione ");
    while(input_state(UP)==1&&input_state(DOWN)==1&& ***
        input_state(OK)==1&&input_state(CANC)==1)
    {}
    if(input_state(OK)==0)
    {
        M_A=1;
        i2c_start();    //Regolazione manuale
        i2c_write(0xA0); // Set Slave address & send data
        i2c_wite(M_A);   // Now send from slave.
        i2c_stop();
        funzione=1;
    }
    if(input_state(UP)==0)
    {funzione=5;}      //Impostazione manuale
    if(input_state(DOWN)==0)
    {funzione=5;}      //Impostazione manuale
    if(input_state(CANC)==0)
    {funzione=1;}      //Ritorno allo standby
}

```

```

void show()
{
    lcd_putc("\f");
    printf(lcd_putc,"Temp int.= %f °C", temp_int);
    lcd_putc("\n");
    printf(lcd_putc,"Temp est.= %f °C", temp_est);
    while(input_state(UP)==1&&input_state(DOWN)==1&& ***

```

```

void temp_select()    //{
{
    lcd_putc("\f");
    lcd_putc("Premere GIU per 35°C");
    lcd_putc("\n");
    lcd_putc("Premere SU per 45°C");
    while(input_state(UP)==1&&input_state(DOWN)==1&& ***
        input_state(OK)==1&&input_state(CANC)==1)
    {}
    if(input_state(OK)==0)
    {funzione=8;}
    if(input_state(UP)==0)
    {
        temp=45;    //Riscaldamento a 45°C
        funzione=1;
    }
    if(input_state(DOWN)==0)
    {
        temp=35;    //Riscaldamento a 35°C
        funzione=1;
    }
    if(input_state(CANC)==0)
    {funzione=1;}    //Ritorno allo standby
}

```

```

void imp_fun()    //Impostazione ventola ON/OFF
{
    lcd_putc("\f");
    lcd_putc("Premere SU per ON ");
    lcd_putc("\n");
    lcd_putc("Premere GIU per OFF");
    while(input_state(UP)==1&&input_state(DOWN)==1&& ***
        input_state(OK)==1&&input_state(CANC)==1)
    {}
    if(input_state(OK)==0)
    {funzione=12;}
    if(input_state(UP)==0) //ventola ON
    {output_high(VENTO);}
    if(input_state(DOWN)==0) //Ventola OFF
    {output_high(VENTO);}
    if(input_state(CANC)==0)
    {funzione=1;}    //Ritorno allo standby
}

```

# Codice sorgente unità SICUREZZA scritto con CCS PICC 4.057

```
#include <Sicurezza1.h>
#include <KBC.c> //Funzione per la gestione del tastierino
//scritta da Marco Di Goro

//Function prototypes
void eeprom();
void allarme();

//Global Defines
#define BUZZ PIN_A6
//PIN collegato al BJT che comanda il BUZZER
#define VERDE PIN_A2
#define ROSSO PIN_A3

//Global Variables
int1 M_A=0;
int state, TASTO, codice=0;
float luce;

#int_SSP //interrupt su "I2C activity"
void SSP_isr(void)
{
    state = i2c_isr_state();

    if(state < 0x80 && state != 0) //master is sending data
    {
        M_A = i2c_read(); //read requested data
    }
    if(state == 0x80) //master is requesting data
    {
        i2c_write(codice); //send requested data
    }
}

void main()
{int duty, V;
    port_b_pullups(TRUE);
    setup_adc_ports(AN0_TO_AN1|VSS_VDD);
    setup_adc(ADC_CLOCK_INTERNAL|ADC_TAD_MUL_2);
    setup_wdt(WDT_OFF);
    setup_timer_0(RTCC_INTERNAL);
    setup_timer_1(T1_DISABLED);
    setup_timer_2(T2_DIV_BY_16,249,1);
    setup_timer_3(T3_DISABLED|T3_DIV_BY_1);
    setup_ccp1(CCP_PWM);
    // set_pwm1_duty(102);
    setup_vref(FALSE);
    enable_interrupts(INT_SSP);
    enable_interrupts(GLOBAL);
    setup_oscillator(OSC_4MHZ|OSC_INTRC|OSC_PLL_OFF);
    eeprom();
    V=1;
    output_high(ROSSO); //Led rosso
    while(V<=8)
    {
        KBC();
        if(TASTO==read_eeprom(V))
        { output_high(VERDE); //Led arancione ad ogni pressione
          output_high(BUZZ); //Suono di conferma pressione tasto
          delay_ms(100);
          output_low(BUZZ);
          delay_ms(150);
          output_low(VERDE);
        }
        else
        {
            allarme();
        }
    }
}
```

```
output_low(ROSSO);
output_high(VERDE); //Led arancione
codice=1;
while(1)
{
    if(M_A==0)
    {set_adc_channel(0);} //Lettura da fotoresistenza
    else
    {set_adc_channel(1);} //Lettura da variatore
    delay_us(10);
    luce=read_adc();
    luce=luce-1;
    luce=luce/1023*100;
    duty=luce*10.23;
    set_pwm1_duty(duty);
    delay_ms(500);
} // {}
}
```

```
void eeprom()
{
    write_eeprom(1,1);
    write_eeprom(2,8);
    write_eeprom(3,0);
    write_eeprom(4,2);
    write_eeprom(5,1);
    write_eeprom(6,9);
    write_eeprom(7,9);
    write_eeprom(8,0);
}
```

```
void allarme()
{
    while(1)
    {
        output_high(BUZZ);
        delay_ms(500);
        output_low(BUZZ);
        delay_ms(250);
    }
}
```

//N.B. I commenti scritti in verde non hanno effetto

