**TITOLO**: Multiplazione di ingressi e uscite

**MATERIE**: Sistemi automatici, TPSEE, Elettronica ed Elettrotecnica

**OBIETTIVI**: sondare le problematiche di acquisizione digitale e di organizzazione logica, in problemi di gestione automatica di sistemi.

**■ ESPOSIZIONE DEL PROBLEMA DI ANALISI/PROGETTAZIONE**

In questo elaborato intendo analizzare come il microcontrollore o un qualsiasi sistema di elaborazione dotato di ingressi e uscite digitali e opportunamente programmato possa gestire il flusso di auto entranti e uscenti da un parcheggio.

Tipicamente il sistema deve rilevare le auto entranti e uscenti attraverso i segnali degli input provenienti dalle fotocellule e in base a questo flusso aggiornare il dato dei posti liberi nella propria memoria. Nel caso non siano più presenti posti liberi deve tipicamente inibire l’accesso di nuovi veicoli e segnalare l’indisponibilità mediante un semaforo rosso.

Si vuole sviluppare un sistema di monitoraggio dello stato di occupazione in un parcheggio con la capacità di 512 autoveicoli suddivisi in otto settori.

Un sistema a microprocessore deve essere in grado di acquisire l’impegno del parcheggio sia in relazione al numero di auto in sosta che alla loro distribuzione nei vari settori. Un opportuno sistema di segnalazione all’ingresso del parcheggio informa gli automobilisti in arrivo indicando per ogni settore la presenza di posti liberi. All’ingresso di ogni settore c’è una sbarra che si chiude automaticamente quando il settore è pieno. All’ingresso principale c’è una sbarra che si chiude quando i posti liberi scendono a 10.

**■ SVILUPPO DELLA SOLUZIONE CON SPIEGAZIONI, SCHEMI, ELABORAZIONI MATEMATICHE**

**■ Schema a blocchi del sistema**

Nello schema implementativo si possono distinguere le seguenti sezioni:

* multiplexer 4/16 per la selezione dei segnali provenienti dalle cellule fotoelettriche in entrata (Si0÷Si7) e in uscita (So0÷So7);
* linee per il comando delle sbarre e dei semafori;
* linee per il comando della sbarra principale e per la segnalazione del parcheggio libero;
* porta di 8 bit collegata direttamente a sbarre e semafori.

**μC**

RA0

RA1

RA2

S0

S1

S2

S3

S4

S5

S6

S7

**Comando apertura/**

**chiusura sbarre**

L0

L1

L2

L3

L4

L5

L6

L7

**Comando segnalazione**

**libero/occupato**

**Comando sbarra principale**

**parcheggio libero**

MULT

I

PLEXER 4/16

sensori auto-IN settore i-esimo

Si0

Si1

Si2

Si3

Si4

Si5

Si6

Si7

So0

So1

So2

So3

So4

So5

So6

So7

RC6

RC0

RA3

RA47

RC1

RC2

RC3

RC4

RC5

RC7

RD0

RD1

sensori auto-OUT settore i-esimo

OUT

L’adozione del multiplexer permette di risparmiare il numero di linee di ingresso, inviando l’indirizzo del dato di ingresso da selezionare.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| I3 | I2 | I1 | I0 |  | U |
| 0 | 0 | 0 | 0 |  | **I0** |
| 0 | 0 | 0 | **1** |  | **I1** |
| 0 | 0 | **1** | 0 |  | **I2** |
| 0 | 0 | **1** | **1** |  | **I3** |
| 0 | **1** | 0 | 0 |  | **I4** |
| 0 | **1** | 0 | **1** |  | **I5** |
| 0 | **1** | **1** | 0 |  | **I6** |
| 0 | **1** | **1** | **1** |  | **I7** |
| **1** | 0 | 0 | 0 |  | **I8** |
| **1** | 0 | 0 | **1** |  | **I9** |
| **1** | 0 | **1** | 0 |  | **I10** |
| **1** | 0 | **1** | **1** |  | **I11** |
| **1** | **1** | 0 | 0 |  | **I12** |
| **1** | **1** | 0 | **1** |  | **I13** |
| **1** | **1** | **1** | 0 |  | **I14** |
| **1** | **1** | **1** | **1** |  | **I15** |

Come si può osservare infatti, a fronte di sedici ingressi, nel microcontrollore perviene il solo ingresso RA4 sulla porta A.

Infatti i livelli degli ingressi vengono analizzati in sequenza differita temporalmente, inviando al MUX l’indirizzo dell’ingresso per mezzo dei segnali RA3RA2RA1RA0 della portA.

È un caso molto istruttivo che ci consente di comprendere come una spesa maggiore a livello software si traduca in un risparmio a livello hardware.

La figura a destra rappresenta la tabella di verità del MUX a quattro ingressi di selezione, sedici ingressi di dati e un’uscita.

Nello schema sono presenti inoltre otto uscite, facenti capo alla PORTC, che in funzione dello stato di occupazione in ogni settore del parcheggio attivano o disattivano il corrispondente semaforo posto all’ingresso del settore. Gli stessi segnali comandano anche la apertura/chiusura delle sbarre.

**■ Algoritmo di gestione**

L’algoritmo esegue i seguenti passi:

1. scandisce in sequenza i sensori auto\_IN e auto\_OUT del settore i-esimo e non appena rileva l’ingresso/uscita di un’auto decrementa/incrementa il conteggio dei posti liberi di tale settore;

se auto\_IN 🡺 allora LIBERI\_i = LIBERI\_i - 1

se auto\_OUT 🡺 allora LIBERI\_i = LIBERI\_i + 1

1. chiude la i-esima sbarra se LIBERI\_i = 0 e accende il corrispondente semaforo rosso;
2. calcola il numero totale di posti liberi TOTLIBERI, sommando LIBERI\_i per 0 < i < 7.

Nel caso si abbia TOTLIBERI = 10 provvede a chiudere la sbarra all’ingresso principale.

E’ necessario dichiarare le seguenti variabili nello spazio della memoria del microcontrollore.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| SI7 | SI6 | SI5 | SI4 | SI3 | SI2 | SI1 | SI0 | SENSORI\_IN |
| FI7 | FI6 | FI5 | FI4 | FI3 | FI2 | FI1 | FI0 | SENSORI\_OUT |

|  |
| --- |
| LIBERI\_0 |
| LIBERI\_1 |
| LIBERI\_2 |
| LIBERI\_3 |
| LIBERI\_4 |
| LIBERI\_5 |
| LIBERI\_6 |
| LIBERI\_7 |
| TOTLIBERI |
| SENSORI\_IN |
| FLAG\_IN |
| SENSORI\_OUT |
| FLAG\_OUT |
| LAMPADE |
| SBARRE |

Si nota la presenza del byte SENSORI\_IN per la memorizzazione dello stato dei sensori di rilevamento del passaggio auto in ciascun settore.

A questo byte si associa FLAG\_IN, mediante il quale si memorizza 1 se si è verificato il passaggio all’ingresso di un’auto.

La logica è la seguente:

* se SIN, FIN di un settore sono entrambi 0 significa che non sta transitando alcuna auto, pertanto non viene fatto nulla;
* se SIN = 1 FIN = 0 significa che l’auto sta transitando, quindi si deve impostare il flag FIN = 1per inibire ogni ulteriore conteggio;
* se SIN = 1 FIN = 1 significa che è stato rilevato il transito dell’auto, che però è ancora in corso;
* se SIN = 0 FIN = 1 significa che l’auto è transitata, quindi si deve decrementare LIBERI\_i e reimpostare FIN = 0.

Lo stesso ragionamento si p

Per quanto riguarda il diagramma di flusso, conviene razionalizzare il problema della stesura suddividendo i compiti del microcontrollore in tre sequenze logiche:

Lettura dati di input (sensori) e creazione immagine in memoria

Elaborazione dati e creazione immagine di memoria delle uscite

Invio dati in output (attuatori)

Per quanto riguarda la sezione di lettura dei dati di input, riporto il diagramma di flusso relativo alla lettura dei 16 sensori (8 in + 8 out). Il diagramma descrive l’invio in sequenza degli indirizzi RA3RA2RA1RA0 al MUX e il prelievo del dato dal pin RA4.

Per quanto riguarda la sezione di elaborazione mi concentro sulla gestione entrate delle auto.

Si noti la presenza di due test. Nel primo si verifica se un’auto sta iniziando a entrare. In caso affermativo viene attivato il corrispondente flag FIN. Solo quando l’auto è transitata interamente (SIN = 0) è possibile decrementare la variabile LIBERI e contestualmente resettare il flag.

Sì

SIN

= 1

No

FIN = 1

Sì

SIN = 0

AND

FIN = 1

No

LIBERI = LIBERI - 1

FIN = 0

I = 0

Collocazione entro

SENSORI\_IN e SENSORI\_OUT

RA3RA2RA1RA0

= I

Lettura RA4

I = I + 1

Sì

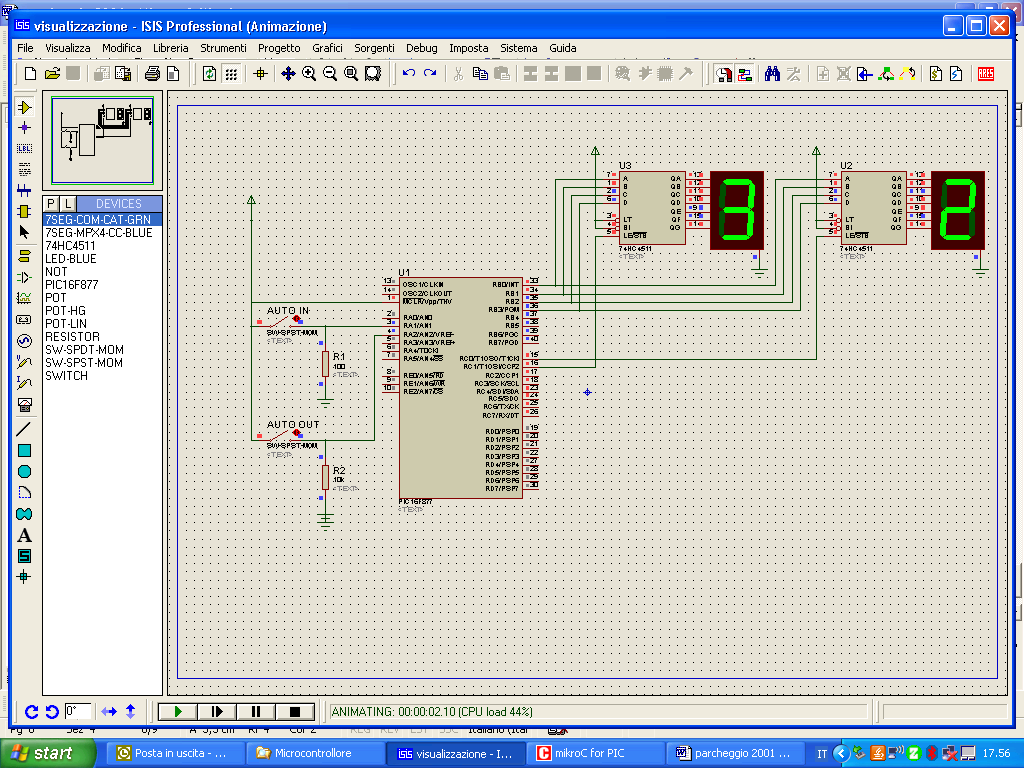
I < 16

No

**■ VALIDAZIONE DEI RISULTATI OTTENUTI CON UNO O PIU’ ESPERIMENTI SIMULATI**

Per saggiare la veridicità dei risultati ho sperimentato un sistema semplificato con una sola sbarra di ingresso e una d’uscita implementato con il programma **Proteus** nel quale:

* RA1 = 1 simula l’entrata di un’auto;
* RA2 = 1 simula l’uscita di un’auto.



Nel caso di auto in entrata per esempio:

1. l’istruzione if (PORTA.F1) che significa “se RA1 = 1” permette di verificare se il pulsante è premuto;
2. la condizione di auto in transito viene memorizzata ponendo flag1 = 1;
3. successivamente, se il pulsante viene rilasciato, ovvero !PORTA.F1 che significa NOT RA1 = 1 ovvero RA1 = 0, viene decrementato il numero di posti liberi LIBERI e viene reinizializzato flag1 = 0.

Nel caso di auto in uscita vale il medesimo ragionamento.

Il programma successivamente converte LIBERI in BCD, separando le componenti UNITA e DECINE.

Queste sono inviate ai rispettivi eventuali display in modo multiplexato, mediante attivazione dei latch.

Una alternativa per rilevare la pressione di un pulsante è:

if (Button(&PORTA, 1, 1, 1)==255)

dove il primo 1 è la posizione del bit nella porta, il secondo 1 è il numero di ms di attesa per antirimbalzo.

//programma incremento/decremento in funzione di pulsanti

//RA1 auto IN; RA2 auto OUT

unsigned int LIBERI, DECINE, UNITA, flag1=0, flag2=0;

void main() {

TRISB = 0; //tutte uscite

TRISC = 0; //tutte uscite

ADCON1 = 0B10001110; // allineamento destra, RA0 analogico,

// RA1, RA2, RA3 digitale, Vref=+Vdd

LIBERI=32;

do {

if (PORTA.F1) flag1=1; //se sensore rileva auto, setta flag

if ((!PORTA.F1)&&(flag1==1)) //se auto entra

{

LIBERI=LIBERI-1; //decrementa posti liberi

flag1=0; //prepara flag per prossima auto

}

if (PORTA.F2) flag2=1;

if ((!PORTA.F2)&&(flag2==1))

{

LIBERI=LIBERI+1;

flag2=0;

}

UNITA=LIBERI%10; //conversione BCD

DECINE=floor(LIBERI/10);

PORTB=UNITA; // visualizzazione

PORTC=0B11111110;

PORTC=0B11111111;

PORTB=DECINE;

PORTC=0B11111101;

PORTC=0B11111111;

} while(1);

}//~

**■ Espansione dell’elaborato mediante contenuti teorici**

Dal manuale di Elettrotecnica, Elettronica e Automazione si può reperire materiale da rielaborare per completare la propria esposizione con parentesi di approfondimento teorico, legate ai contenuti dell’elaborato.

* Sezione XIII Acquisizione dati e interfacciamento – Acquisizione e distribuzione dati, conversione analogico digitale, interfacciamento.
* Sezione XIV Microcontrollori – Programmazione dei microcontrollori, porte di input-output.

**■ Suggerimenti di possibili percorsi alternativi per nuovi elaborati.**

* Sistemi con multiplexing degli ingressi, delle uscite e dei dati trasmessi.
* Confronto tra conteggio di eventi (come le auto che entrano e escono) in modalità polling (come in questo elaborato) e in modalità interrupt.