



Ministero dell'Istruzione, dell'Università e della Ricerca

M048 - ESAME DI STATO DI ISTITUTO PROFESSIONALE

CORSO DI ORDINAMENTO

Indirizzo: TECNICO DELLE INDUSTRIE ELETTRONICHE

Tema di: SISTEMI, AUTOMAZIONE E ORGANIZZAZIONE DELLA PRODUZIONE

Un sistema a microprocessore controlla un dispositivo automatico di apertura e chiusura di un cancello, mediante l'impiego di porte di ingresso e di uscita opportunamente interfacciate.

Il sistema si compone essenzialmente di:

- a) un motore, che può ruotare nei due sensi per consentire l'apertura o la chiusura del cancello;
- b) due interruttori di fine corsa, che si chiudono rispettivamente a cancello completamente aperto e completamente chiuso;
- c) un sensore di prossimità che segnala la presenza di qualcuno nei pressi del cancello.

Il candidato, formulate le dovute ipotesi aggiuntive, determini:

- 1) un diagramma a blocchi che descriva l'intero sistema;
- 2) diagrammi dettagliati illustranti le soluzioni proposte per l'interfacciamento di sensori e attuatori;
- 3) lo schema di utilizzo delle porte di I/O impiegate;
- 4) un diagramma degli stati che descriva il funzionamento del sistema, o in sua alternativa, un diagramma di flusso di tipo funzionale per il programma di controllo del sistema;
- 5) la codifica del software di controllo in un linguaggio di programmazione di sua conoscenza.

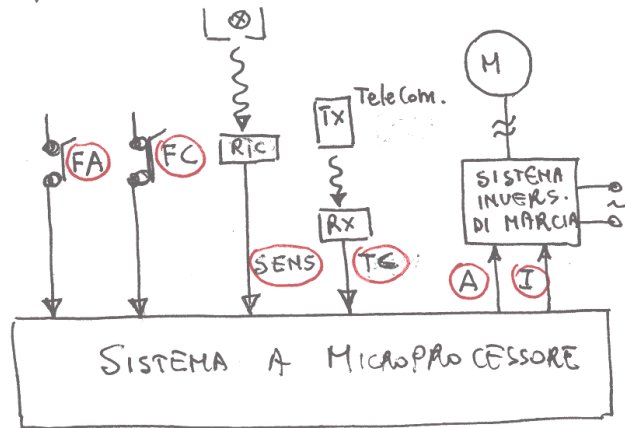
Durata massima della prova: 6 ore

E' consentito soltanto l'uso di manuali tecnici e di calcolatrici tascabili non programmabili.
Non è consentito lasciare l'Istituto prima che siano trascorse 3 ore dalla dettatura del tema.

SOLUZIONE

1

1) Si propone il seguente sistema



SEGNALI I/O

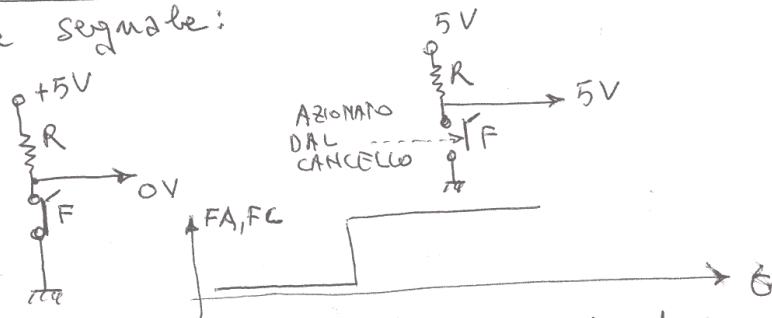
FA : fine corsa aperto
 FC : fine corsa chiuso
 SENS : sensore prossimità
 TC : telecomando
 A : comanda avanti
 I : comanda indietro

I segnali sono attivi
 a livello logico '1'
 e non attivi a livello
 logico '0'

Si prevede l'aggiunta di un telecomando a infrarossi che invia impulsi di comando per aprire e chiudere il cancello a distanza.

Si nota che i segnali di ingresso sono di due tipi: a impulsi, sensore di prossimità e telecomando, e di tipo a livello, FA e FC.

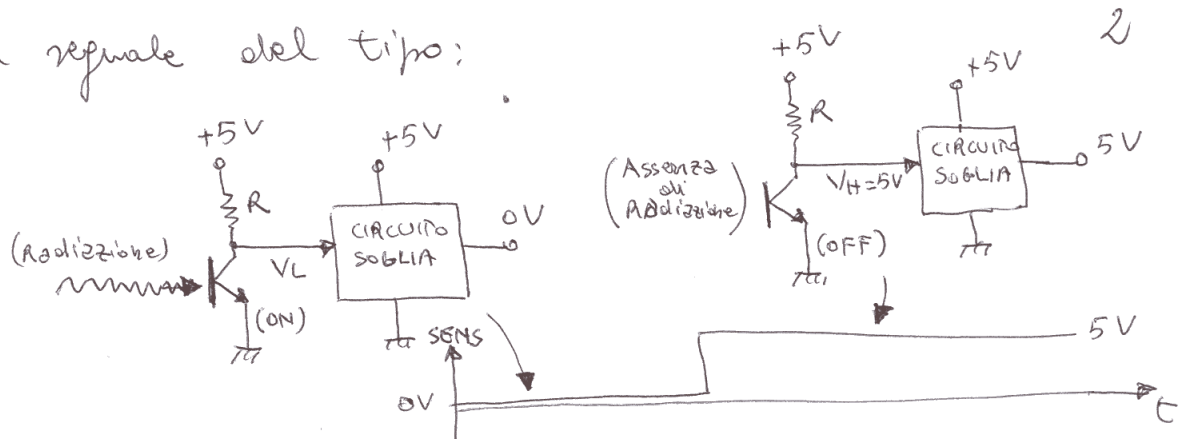
2) I due finecorsa FA ed FC, chiusi a riposo, producono il seguente segnale:



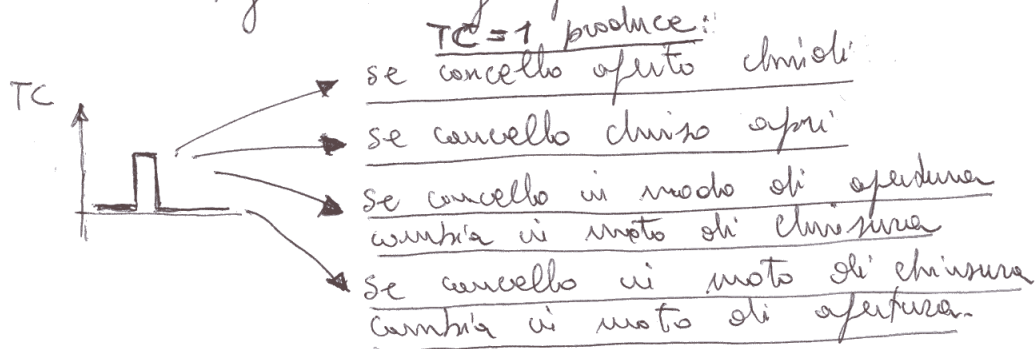
Per cui $FA=0$ $FC=1$ significa cancello chiuso, $FA=1$ $FC=0$ significa cancello aperto, $FA=FC=0$ significa cancello in movimento (in apertura o in chiusura)

Il sensore di prossimità, formato da un emettitore a raggi infrarossi (TX) e da un rivelatore (RX) con elemento fotosensibile come fototransistor o fotodiodo fornisce

un segnale del tipo:



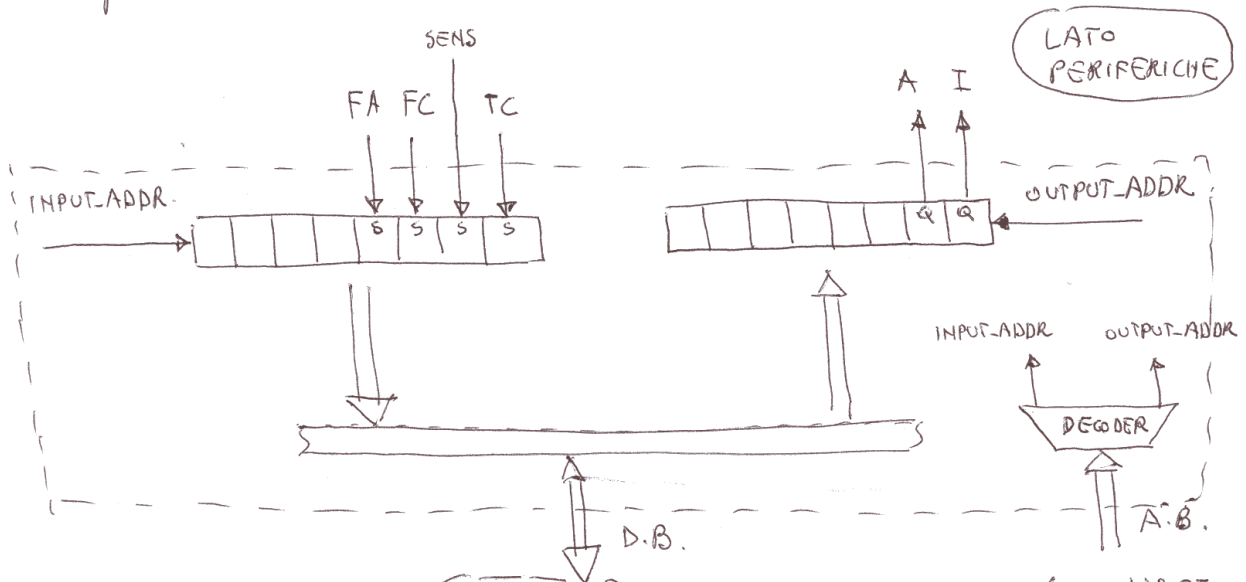
Il telecomando, ad ogni attivazione di un pulsante, invia un codice, formato da una successione di simboli '1' e '0' che modulano una portante di alta frequenza (infrarosso). La successione di questi simboli, ha la funzione di una chiave che, demodolata dal ricevitore Rx, viene confrontata con la chiave memorizzata nel ricevitore. In caso di uguaglianza tra chiave trasmessa e chiave locale del ricevitore viene prodotto un impulso al quale il microcontrollore dà il seguente significato -



Infine l'attuatore, formato da un motore a induzione monofase, è pilotato da un sistema a inversione di marcia sul quale agiscono i segnali di comando A e I che svolgono le seguenti azioni:

A=0	I=0	motore fermo
A=1	I=0	movimento in avanti (apertura)
A=0	I=1	movimento indietro (chiusura)

3) Le porte di I/O integrate, del tipo parallelo, possono essere ricondotte allo schema generale mostrato in figura

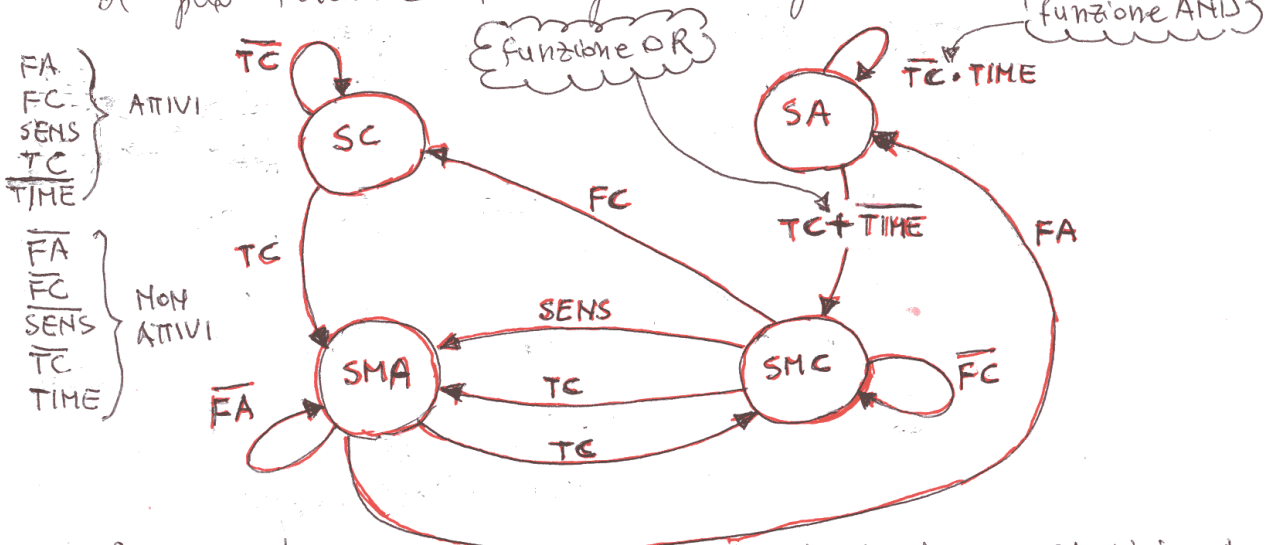


Integrati che dispongono di questa logica sono, ad esempio, il VIA (86522) il PIO (8255).

4) Il sistema potrebbe gestire stati interni

- SC** : stato cancello chiuso
- SA** : stato cancello aperto
- SMC** : stato moto in chiusura
- SMA** : stato moto in apertura

Si può tracciare il seguente diagramma di stato



NB: Per ogni transizione sono stati indicati solo i segnali di input che la determinano (gli altri hanno valore di facile determinazione).

4

Si prevede che lo stato di cancello aperto (SA) sia instabile, per cui, dopo un tempo pari a 2 minuti = 120 sec (se in questo intervallo di tempo non è stata comandata una chiusura con TC attivo), il cancello si richiuderà automaticamente.

Per ottenere questa funzionalità è previsto l'uso di una variabile TIME, inizializzata a 120 allo start della temporizzazione e decrementata di uno, ad ogni secondo, dalla risposta ad un evento timer opportunamente programmato (interrupt).

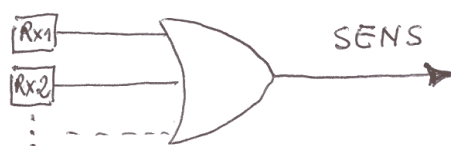
Il valore zero della variabile TIME decreterà la fine della temporizzazione di due minuti.

Nel flow-chart con il simbolo TIME si intende il valore 0 di questa variabile, che corrisponde al valore attivo. Ovvero il simbolo TIME ($\neq 0$, valore tra 120 e 1) indica il valore non attivo.

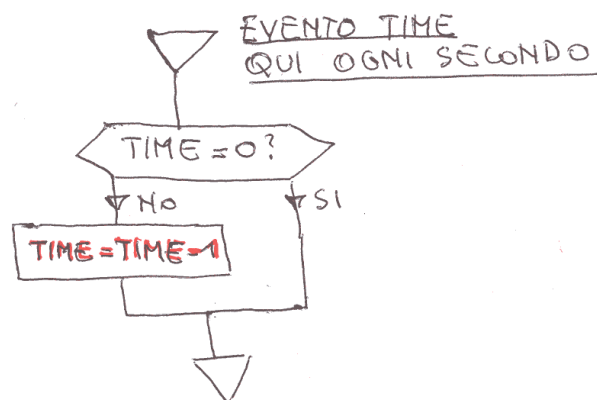
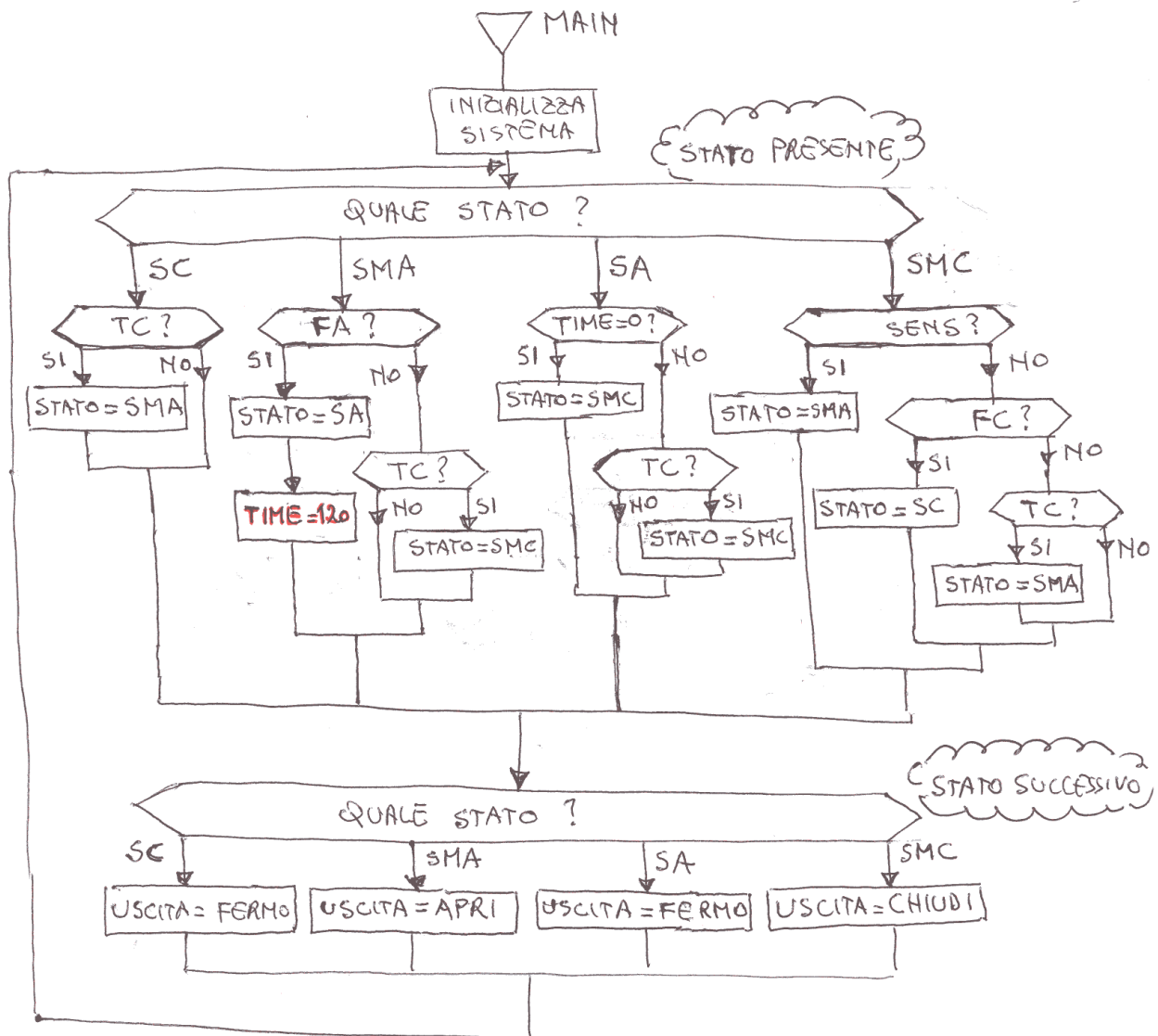
La transizione tra lo stato SA e SMC è prodotta dal segnale di ingresso TC + TIME che significa TC attivo (=1) o TIME attivo (=0) o entrambi. La permanenza nello stato SA avviene quando TC non è attivo (=0) e TIME non è attivo.

In fine si osserva che il segnale SENS, che produce la transizione tra gli stati SMC e SMA, diventa attivo (=1) quando il sensore di prossimità rileva la presenza di un ostacolo.

Questa stessa transizione deve essere prodotta nel caso vi siano due o più sensori distribuiti ai perimetri del cancello. Si può pensare ad un collegamento in OR delle uscite dei diversi ricevitori Rx



Il flow-chart del programma principale e della risposta all'evento TIME sono di seguito riportati: 5



5) Il programma che realizza il controllo, scritto in 6
Visual BASIC può, in linea di massima, avere la
seguente forma

```

CALL INIZIALIZZA 'Definisce e inizializza variabile: FA, FC, SENS, TC, TIME
                  'e costanti: SC, SMA, SA, SMC, FERMO, APRI, CHIUDI
DO
  SELECT CASE STATO 'Determina la transizione dello stato
    CASE SC
      IF TC = 1 THEN
        STATO = SMA 'Inizia apertura
      END IF
    CASE SMA
      IF FA = 1 THEN
        STATO = SA 'Cancello aperto
        TIME = 120 'Start temporizzazione se aperto
      ELSE
        IF TC = 1 THEN STATO = SMC 'Inverti - Chiudi
      END IF
    CASE SA
      IF TIME = 0 THEN
        STATO = SMC 'Richiusura ou 2 minuti di aperto
      ELSE
        IF TC = 1 THEN STATO = SMC 'Inizia chiusura
      END IF
    CASE SMC
      IF SENS = 1 THEN
        STATO = SMA 'Ostacolo! Riapri
      ELSE
        IF FC = 1 THEN
          STATO = SC 'Cancello chiuso
        ELSE
          IF TC = 1 THEN STATO = SMA 'Inverti - Apri
        END IF
      END IF
    END SELECT
  SELECT CASE STATO 'Nuova uscita
    CASE SC
      USCITA = FERMO
    CASE SMA
      USCITA = APRI
    CASE SA
      USCITA = FERMO
    CASE SMC
      USCITA = CHIUDI
  END SELECT
LOOP

```

```

' QUI OGNI SECONDO
SUB EVENTO-TIMER
  IF TIME <> 0 THEN
    TIME = TIME - 1
  END IF
END SUB

```

Lo stesso programma scritto in C#, con l'uso della programmazione a oggetti, può assumere la seguente forma:

```
// Programma in C per PIC18F452
```

```
/*  
  Controllo Cannello  
*/
```

```
short time = 5;  
short dcmsec = 10;
```

```
void interrupt () {  
  if(PIR1.TMR1IF) {  
    TMR1H=0x0B; //20MHz/4=5MHz, 5MHz/8=625000Hz, T=1/625000=1.6 microsec  
    TMR1L=0xDC; //(modulo) 62500*1.6=0.1 sec 65536-62500=3036=0x0BDC  
    dcmsec--;  
    if(dcmsec == 0) {  
      PORTB = PORTB ^ 0x80; //test interrupt  
      dcmsec = 10;  
      if (time != 0) time-- ;  
    }  
  }  
  PIR1.TMR1IF = 0;  
}
```

```
void initTimer1() {  
  INTCON.GIE = 1; // Abilita interrupt globale  
  INTCON.PEIE = 1; // Abilita interrupt periferiche  
  T1CON.T1CKPS1 = 1; // Prescaler 1:8  
  T1CON.T1CKPS0 = 1;  
  T1CON.TMR1CS = 0; //Clock interno (Fosc4)  
  PIR1.TMR1IF = 0; //Reset bit si stato TIMER1. Posto a 1 da interrupt  
  PIE1.TMR1IE = 1; //Abilita interrupt TIMER1  
  T1CON.TMR1ON = 1; //Abilita Timer1  
  TMR1H=0x0B; //20MHz/4=5MHz, 5MHz/8=625000Hz, T=1/625000=1.6 microsec  
  TMR1L=0xDC; //(modulo) 62500*1.6=0.1 sec 65536-62500=3036=0x0BDC  
}
```

```
void main(){  
  
  enum {SC, SMA, SA, SMC} stato = SC;  
  enum {OFF = 0, CLOSE = 1, OPEN = 3} uscita = OFF;  
  
  short FA = 0x02;short FC=0x01;short SENS=0x10;short TC=0x04;  
  
  initTimer1();  
  
  PORTB = 0x00;  
  TRISB = 0x00;  
  TRISC = 0xFF;  
  
  while (1) {  
  
    Delay_ms(100); // Aspetta rimbalzi 100 msec  
    // In funzione dello stato presente e degli ingressi attuali, letti da  
    // PORTC, determina il nuovo stato  
    switch (stato) {  
      case SC:  
        if(PORTC & TC) {  
          stato = SMA;  
          while(PORTC&TC); //aspetta rilascio  
        }  
        break;
```

```

case SMA:
    if(PORTC & FA) {
        stato = SA;
        time = 10;
    }
    else {
        if(PORTC & TC) {
            stato = SMC;
            while(PORTC&TC); //aspetta rilascio
        }
    }
    break;
case SA:
    if(time == 0) {
        stato = SMC;
    }
    else {
        if(PORTC & TC) {
            stato = SMC;
            while(PORTC&TC); //aspetta rilascio
        }
    }
    break;
case SMC:
    if(PORTC & SENS) {
        stato = SMA;
        while(PORTC&SENS); //aspetta rilascio
    }
    else {
        if(PORTC & FC) {
            stato = SC;
        }
        else {
            if(PORTC & TC) {
                stato = SMA;
                while(PORTC&TC){;} //aspetta rilascio
            }
        }
    }
    break;
}

```

// Con il nuovo stato pilota l'uscita

```

switch (stato) {
case SC:
    uscita = OFF;
    break;
case SMA:
    uscita = OPEN;
    break;
case SA:
    uscita = OFF;
    break;
case SMC:
    uscita = CLOSE;
    break;
}
PORTB = uscita;

```

```

}
}

```

//Programma in C# con uso di programmazione a oggetti

```
using System;
using System.Timers;
using System.IO;

namespace Cancellor
{
    enum vS : byte
    {
        SC, SMA, SA, SMC
    }

    public class autGate
    {
        #region Definisce il porto di comunicazione (seriale) con la scheda
        //di interfaccia ai sensori,...
        //definisce i valori da inviare in uscita e definisce le
        //varibili associate agli ingressi
        //e definisce la variabile di temporizzazione
        public MSCommLib.MSCommClass myPort = new MSCommLib.MSCommClass();
        public byte FERMO = 0;
        public byte APRI = 2;
        public byte CHIUDI = 1;
        public byte fa, fc, sens, tc;
        public byte time;
        #endregion

        public autGate()
        {
            this.myPort.CommPort=1;
            this.myPort.Settings="9600,N,8,1";
            this.myPort.PortOpen=true;
            this.myPort.InputLen = 0; //Legge tutto il buffer
            this.myPort.RThreshold=0; //Disabilita l'evento OnComm
        }

        public void readInput()
        {
            byte inByte;
            inByte = Convert.ToByte(this.myPort.Input); //Leggi stato dei
            sensori

            fa = Convert.ToByte(inByte & 0x08);
            fc = Convert.ToByte(inByte & 0x04);
            sens = Convert.ToByte(inByte & 0x02);
            tc = Convert.ToByte(inByte & 0x01);
        }

        public byte newStato(byte s)
        {
            byte stato = s;
            switch ((vS)s)
            {
                case vS.SC:
                    stato = (byte)vS.SC;
                    if (tc == 1) stato = (byte)vS.SMA; //apri
                    break;
                case vS.SMA:
                    stato = (byte)vS.SMA;
            }
        }
    }
}
```

chiudi

```
        if (fa == 1)
        {
            stato = (byte)vS.SA; //aperto
            time = 120;
        }
        else
        {
            if (tc == 1) stato = (byte)vS.SMC; //chiudi
        }
        break;
    case vS.SA:
        stato = (byte)vS.SA;
        if (time == 0)
        {
            stato = (byte)vS.SMC; //passati 2 min,
        }
        else
        {
            if (tc == 1) stato = (byte)vS.SMC; //chiudi
        }
        break;
    case vS.SMC:
        stato = (byte)vS.SMC;
        if (fc == 1)
        {
            stato = (byte)vS.SC; //chiuso
        }
        else
        {
            if (tc == 1) stato = (byte)vS.SMA; //chiudi
        }
        break;
    }
    return stato;
}

public void newUscita(byte s)
{
    byte uscita;
    switch ((vS)s)
    {
        case vS.SC:
            uscita = FERMO;
            break;
        case vS.SMA:
            uscita = APRI;
            break;
        case vS.SA:
            uscita = FERMO;
            break;
        case vS.SMC:
            uscita = CHIUDI;
            break;
        default:
            uscita = FERMO;
            break;
    }
    myPort.Output = Convert.ToString(uscita) ; //invia la
                                                //variabile uscita al porto di uscita
}

}
```

```

class myApp
{
    static autGate mioCancello = new autGate();
    static void Main()
    {
        Timer timeGate = new Timer();
        timeGate.Elapsed += new ElapsedEventHandler(timeGate_Elapsed);
        timeGate.Interval = 1000; //time-out dopo un secondo
        timeGate.AutoReset = true; //deve generare l'evento ogni secondo
        timeGate.Start(); //Avvia la temporizzazione

        byte stato = (byte)vS.SC; //Assumi stato chiuso
        byte uscita = mioCancello.FERMO; //Assumi cancello fermo

        do
        {
            mioCancello.readInput();
            stato = mioCancello.newStato(stato);
            mioCancello.newUscita(stato);
        }while (true);
    }
    public static void timeGate_Elapsed(object source, ElapsedEventArgs e)
    {
        if (mioCancello.time != 0) mioCancello.time--;
    }
}

```