Firmware del sistema di acquisizione automatico da Cosmic Box, presentato al RCM del 25/3/2026
Liceo Scientifico “Corradino D'Ascanio” - Montesilvano.

La scheda utilizzata è una “Arduino 101”

Da mettere a punto la funzione media (comando M)

Per info e supporto, **christian.lenaz@liceodascanio.edu.it**

```
// automazione delle acquisizioni con scheda Arduino a partire dall'intercettazione dei segnali in uscita dalla Cosmic Box: conteggio eventi, ritardo tra eventi successivi
// i comandi giungono dalla UART; i risultati vengono spediti sulla UART
// programma 1 - acquisizione sequenza ritardi: Snn -  S40 = acquisisci e memorizza 40 ritardi (in ms)
// programma 2 - conteggio numero eventi in tot secondi: Ennnn  E0900 = conta numero eventi in un quarto d'ora  -   parametro compreso tra 1 e 9999
// programma 3 - conteggio numero eventi in tot secondi, per più volte: Ennnn,mm - E3600,24 = conta numero eventi in un'ora, per 24 volte
// programma 4 - ritardo medio:  Mnnnn - M1200 = ritardo medio in 1200 secondi
// programma 5 - ritardo medio, per più volte:  Mnnnn,mm - M1200,10 = ritardo medio in 1200 secondi, per 10 volte
// programma 6 - restituzione valori:  REPORT   restituisce un numero (come in E3600 o in M1200) o  più numeri (40 per S40; 24 per E3600,24; 10 per M1200,10 )
// programma 7 - RESET - reset del sistema e preparazione al comando successivo
// finchè non termina l'esecuzione di un comando, il sistema non accetta altri comandi, nemmeno REPORT o RESET

const byte interruptPin=2;
const byte LedRosso=12;
const byte LedVerde=8;
const int MAX_VALORI_DA_RESTITUIRE=99; // numero massimo di valori da restituire in UART
const int MAX_LUNGHEZZA_STRINGA_COMANDO=10; // massima lunghezza stringa di comando
// const byte PinSelezioneProgramma=5; NON SERVE PIU, IL PROGRAMMA DA ESEGUIRE GIUNGE DA UART
//const int Periodo =15000; // conta circa 15 secondi (millisecondi*1000)
int ricevuto=0;
int contatore=0; // numero eventi registrati
int ritardo=0; // millisecondi trascorsi dall'evento precedente
int tempo=0; // milisecondi dal reset
byte LedFlag=0; // attivazione alternata dei due Led, nel programma 2
//String comando; // stringa ricevuta da UART
int durata_sequenza; // numero eventi o durata in secondi da trattare
int numero_iterazioni; // numero di ripetizioni del comando base////////////////////////////////
int i; // variabile di conteggio
String stringa_report; // stringa di commento, mandata in UART prima dei dati numerici
int valori_da_restituire[MAX_VALORI_DA_RESTITUIRE]; // array in cui vengono salvati
```

```
i valori da riversare successivamente in UART
int valori_acquisiti; // numero valori acquisiti
int num_valori_da_restituire; // numero di valori da restituire, in base
all'elaborazione appena effettuata
char comando[MAX_LUNGHEZZA_STRINGA_COMANDO]; // stringa ricevuta da UART
int num_bytes; // numero di bytes letti da seriale, da interpretare come comando
ricevuto
char inByte; // variabile in cui depositare i bytes letti dal buffer seriale,
durante la pulizia dello stesso
int somma; // somma ritardi, per il calcolo della media
int media; // valore medio dei ritardi in tot tempo, in ms
int programma_da_eseguire=0; // programma da eseguire: 1 - 2 - 3 - 4 - 5 - 6 - 7

void isr() { // eseguita ogni volta che arriva un impulso dalla Cosmic Box
 ricevuto=1;
 delay(30);  // serve solo per antirimbalzo in modalità manuale; togliere in caso
di vero interrupt da Cosmic Box
 };


void programma1() { // acquisisce il ritardo tra un impulso e il successivo, e
memorizza in valori_da_restituire
 valori_acquisiti=0; contatore=0;
 stringa_report="Ritardo in millisecondi tra " + (String) (durata_sequenza) + "
eventi successivi: ";
 num_valori_da_restituire=durata_sequenza; ricevuto=0;
 //while (!ricevuto) { } ;  // attesa primo interrupt
 ritardo=0;
 while (valori_acquisiti<num_valori_da_restituire) {
  delay(1); // attendi un millisecondo
  ritardo++;
  // tempo++;
  if(ricevuto) {
   valori_da_restituire[valori_acquisiti] = ritardo;
   valori_acquisiti++;
   contatore++;  // per pilotaggio LED
   ricevuto=0; ritardo=0;
   if (contatore%2==0) {
    digitalWrite(LedRosso,HIGH); digitalWrite(LedVerde,LOW); }
   else {
    digitalWrite(LedRosso,LOW); digitalWrite(LedVerde,HIGH); };
   } // fine 'ricevuto'
  }  // fine ciclo while
}   // fine programma1

void programma2() { // conta gli eventi in durata_sequenza secondi
 programma3();  // programma3 ingloba anche programma2, nel caso particolare in cui
numero_iterazioni=1;
}    // fine programma2
```

```
void programma3() {  // conta gli eventi in durata_sequenza secondi, per
numero_iterazioni volte
 stringa_report="Numero eventi in  " + (String) (durata_sequenza) + " secondi";
 if (numero_iterazioni>1) stringa_report=stringa_report + ", per " +
(String)numero_iterazioni + " volte: ";
  else  stringa_report=stringa_report +": ";
 num_valori_da_restituire=numero_iterazioni;   //Serial.print(" durata= "+
(String)durata_sequenza+" iterazioni= "+(String)numero_iterazioni );//delay(5000);
 for (i=0; i<numero_iterazioni; i++) {
  tempo=0; contatore=0; valori_acquisiti=0; LedFlag=0;
  while ( tempo < (durata_sequenza*1000) ) {
   delay(1); // attendi un millisecondo
   tempo++;
   if(ricevuto) {
    contatore++;
    ricevuto=0;
    if (LedFlag) {
     digitalWrite(LedRosso,HIGH); digitalWrite(LedVerde,LOW); LedFlag=0; }
    else {
     digitalWrite(LedRosso,LOW); digitalWrite(LedVerde,HIGH); LedFlag=1; }
    //} // fine LedFlag
   }  // fine 'ricevuto'
  };  // fine ciclo while
  valori_da_restituire[i] = contatore;
 }   // fine ciclo for
}    // fine programma3


void programma5() { // calcola la media dei ritardi in durata_sequenza secondi, per
numero_iterazioni volte
 stringa_report="Ritardo medio in  " + (String)(durata_sequenza) + " secondi: ";
 if (numero_iterazioni>1) stringa_report=stringa_report + " ,per " +
int(numero_iterazioni) + " volte.";
 num_valori_da_restituire=numero_iterazioni;
 valori_acquisiti=0; ricevuto=0;

 for (i=0; i<numero_iterazioni; i++) {
  tempo=0; contatore=0; somma=0;
  while (tempo<durata_sequenza) {
   delay(1); // attendi un millisecondo
   tempo++;
   if(ricevuto) {
    somma+=tempo;
    contatore++;
    ricevuto=0;
    if (LedFlag) {
     digitalWrite(LedRosso,HIGH); digitalWrite(LedVerde,LOW); LedFlag=0; }
    else {
     digitalWrite(LedRosso,LOW); digitalWrite(LedVerde,HIGH); LedFlag=1; }
   }  // fine 'ricevuto'
  }  // fine while (tempo.....)
```

```
 media=int(somma/contatore);
 valori_da_restituire[valori_acquisiti] = media;
 valori_acquisiti++;
 }   // fine ciclo for
}    // fine programma5


void programma6() { // restituisce i dati elaborati
 Serial.print(stringa_report);
 if(num_valori_da_restituire>0) Serial.print (valori_da_restituire[0]);
 for (i=1; i<num_valori_da_restituire; i++) {
  Serial.print (','); Serial.print (valori_da_restituire[i]);
  }
 Serial.println(" ");
 }

void programma7() { // reset del sistema
 Serial.println("RESET DEL SISTEMA");
 num_valori_da_restituire=0;
 while (Serial.available())   // pulizia buffer UART
   int inByte = Serial.read();
 }

void setup() {
 pinMode(LedRosso,OUTPUT); // LED rosso
 digitalWrite(LedRosso,LOW);
 pinMode(LedVerde,OUTPUT); // LED verde
 digitalWrite(LedVerde,LOW);
 //pinMode(PinSelezioneProgramma,INPUT); // selezione programma di funzionamento:
ground=intervalli di tempo, Vcc=impulsi in 15s

 pinMode(interruptPin,INPUT_PULLUP);
 attachInterrupt(digitalPinToInterrupt(interruptPin), isr, RISING);
 Serial.begin(9600);
 digitalWrite(LedRosso,HIGH); digitalWrite(LedVerde,HIGH);delay(2000);
digitalWrite(LedRosso,LOW); digitalWrite(LedVerde,LOW);delay(2000);

 Serial.println("Sistema pronto. Inserisci una stringa di comando");
 menu();
}

int valore( char val) {
 if (val=='0') return 0;
 if (val=='1') return 1;
 if (val=='2') return 2;
 if (val=='3') return 3;
 if (val=='4') return 4;
 if (val=='5') return 5;
 if (val=='6') return 6;
 if (val=='7') return 7;
 if (val=='8') return 8;
```

```
 if (val=='9') return 9;
 } // end valore



void menu()  {  // manda su seriale il menu dei comandi
 Serial.println("Snn - acquisisci e memorizza nn ritardi.");
 Serial.println("Ennnn - conta numero eventi in nnnn secondi.");
 Serial.println("Ennnn,mm - conta numero eventi in nnnn secondi, per mm volte.");
 Serial.println("Mnnnn - ritardo medio in millisecondi, in nnnn secondi.");
 Serial.println("Mnnnn,mm - ritardo medio in millisecondi, in nnnn secondi, per mm
volte.");
 Serial.println("REPORT - restiuisce il risultato dell'ultimo comando eseguito.");
 Serial.println("RESET - reset del sistema.");
} // fine menu


void loop() {
 programma_da_eseguire=0; durata_sequenza=0;
 if (Serial.available()==0) {  delay(400); } // se non è arrivata una stringa, non
fare nulla
 else {
  for (int j=0;j<MAX_LUNGHEZZA_STRINGA_COMANDO;j++) { comando[j]= ' '; }; // resetta
l'array comando, che ospiterà il prossimo comando da eseguire
  //programma_da_eseguire=0; durata_sequenza=0;
  num_bytes=0;
  while (Serial.available()>0) {
   comando[num_bytes]=Serial.read();
   num_bytes++;
   } // fine ciclo while di lettura del buffer seriale


  if (comando[0]=='S') { // può essere un comando del tipo Snnn - durata_sequenza
contiene il numero di eventi da acquisire e memorizzare
   durata_sequenza=0;
   if ( isDigit(comando[1]) && isDigit(comando[2]) ) {
    durata_sequenza = (valore(comando[1])*10) + valore(comando[2]);

     //if(isDigit(comando[3])) {  // **terza cifra disattivata per scarsa memoria
RAM su Arduino Uno**
     // durata_sequenza*=10;
     // durata_sequenza_temp+=int(comando[3]);
     // }
     // else durata_sequenza=0;
     //}
     //else durata_sequenza=0;
    }
   if (durata_sequenza>0) programma_da_eseguire=1;
   } // fine comando[0]=='S'

  if (comando[0]=='E') { // può essere un comando del tipo Ennnn o Ennnn,mm -
```

```
            //durata_sequenza contiene il periodo di tempo da monitorare, in
secondi;
            //numero_iterazioni contiene le ripetizioni da effettuare
            // la sequenza Ennnn seguita da qualcosa di non identificabile con ,mm
viene interpretata come Ennnn
  durata_sequenza=0; // se il comando è corretto, conterrà la durata in secondi di
ciascun periodo da monitorare
  numero_iterazioni=1; // se il comando è corretto, conterrà il numero di periodi
da considerare
  if( isDigit(comando[1]) && isDigit(comando[2]) && isDigit(comando[3]) &&
isDigit(comando[4]) ) {
   durata_sequenza= (valore(comando[1])*1000) + (valore(comando[2])*100) +
(valore(comando[3])*10) + valore(comando[4]);
   if ( (comando[5]==',') && isDigit(comando[6]) && isDigit(comando[7]) ) {
    numero_iterazioni = (valore(comando[6])*10) + valore(comando[7]);
    };
   };
  if ((durata_sequenza>0) && (numero_iterazioni==1)) programma_da_eseguire=2;
  if ((durata_sequenza>0) && (numero_iterazioni>1)) programma_da_eseguire=3;
  } // fine comando[0]=='E'


 if (comando[0]=='M') { // può essere un comando del tipo Mnnnn o Mnnnn,mm -
durata_sequenza contiene il periodo di tempo da monitorare, in secondi;
numero_iterazioni contiene le ripetizioni da effettuare
            // la sequenza Ennnn seguita da qualcosa di non identificabile con ,mm
viene interpretata come Ennnn
  durata_sequenza=0;    // se il comando è corretto, conterrà la durata in secondi
di ciascun periodo da monitorare
  numero_iterazioni=1;  // se il comando è corretto, conterrà il numero di periodi
da considerare
  if( isDigit(comando[1]) && isDigit(comando[2]) && isDigit(comando[3]) &&
isDigit(comando[4]) )  {
   durata_sequenza= (valore(comando[1])*1000) + (valore(comando[2])*100) +
(valore(comando[3])*10) + valore(comando[4]);
   if ( (comando[5]==',') && isDigit(comando[6]) && isDigit(comando[7]) ) {
    numero_iterazioni = (valore(comando[6])*10) + valore(comando[7]);
    }
   }
  if ((durata_sequenza>0) && (numero_iterazioni==1)) programma_da_eseguire=4;
  if ((durata_sequenza>0) && (numero_iterazioni>1)) programma_da_eseguire=5;
  } // fine comando[0]=='M'

 if ((comando[0]=='R') && (comando[1]=='E') && (comando[2]=='P') &&
(comando[3]=='O') && (comando[4]=='R') && (comando[5]=='T') )  // comando di
restituzione dati elaborati
  programma_da_eseguire=6;

 if ((comando[0]=='R') && (comando[1]=='E') && (comando[2]=='S') &&
(comando[3]=='E') && (comando[4]=='T'))  // comando di reset
  programma_da_eseguire=7;
```

```
 Serial.print("Comando ricevuto: "); for (int j=0; j<MAX_LUNGHEZZA_STRINGA_COMANDO;
j++) { Serial.print(comando[j]);};
 if (programma_da_eseguire==0) { Serial.println("Comando errato."); menu(); } else
Serial.println("Eseguo...");
 Serial.println(" ");

 if (programma_da_eseguire==1) programma1();
 if ( (programma_da_eseguire==2) || (programma_da_eseguire==3) ) programma3();  //
programma3 ingloba anche programma2, nel caso particolare in cui
numero_iterazioni=1;
 if ( (programma_da_eseguire==4) || (programma_da_eseguire==5) ) programma5();  //
programma5 ingloba anche programma4, nel caso particolare in cui
numero_iterazioni=1;
 if (programma_da_eseguire==6) programma6();  // restituzione dati elaborati
 if (programma_da_eseguire==7) programma7();  // reset
 }  // fine Serial.available
}  // fine loop
```