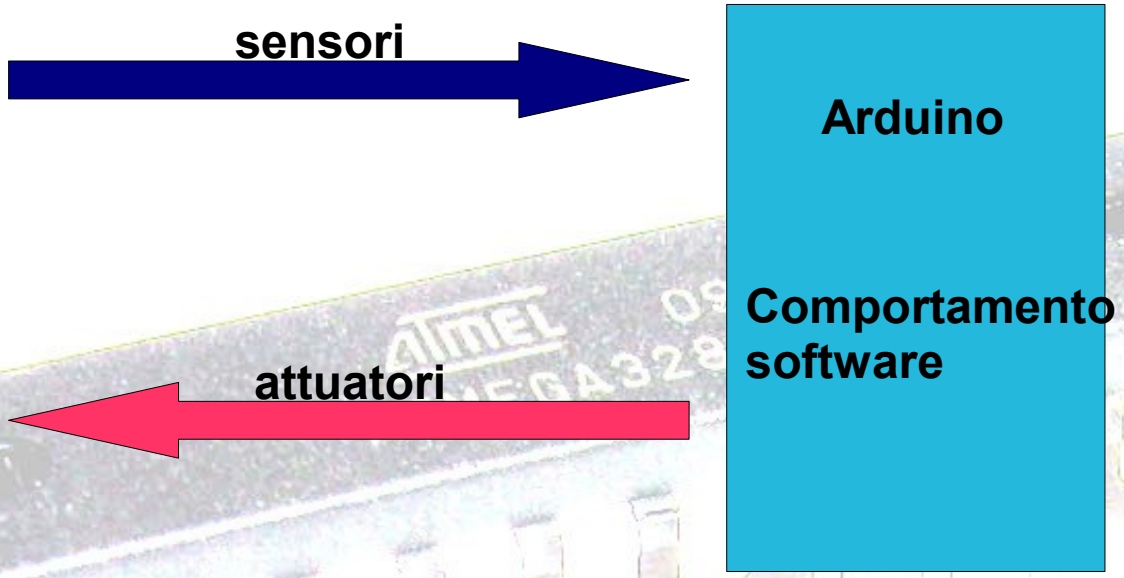
PROGETTI CON ARDUINO UNO

-Introduzione alla scheda Arduino-

www.arduino.cc
sistemisds.altervista.org

Novembre 2013 - Sergio De Sordi

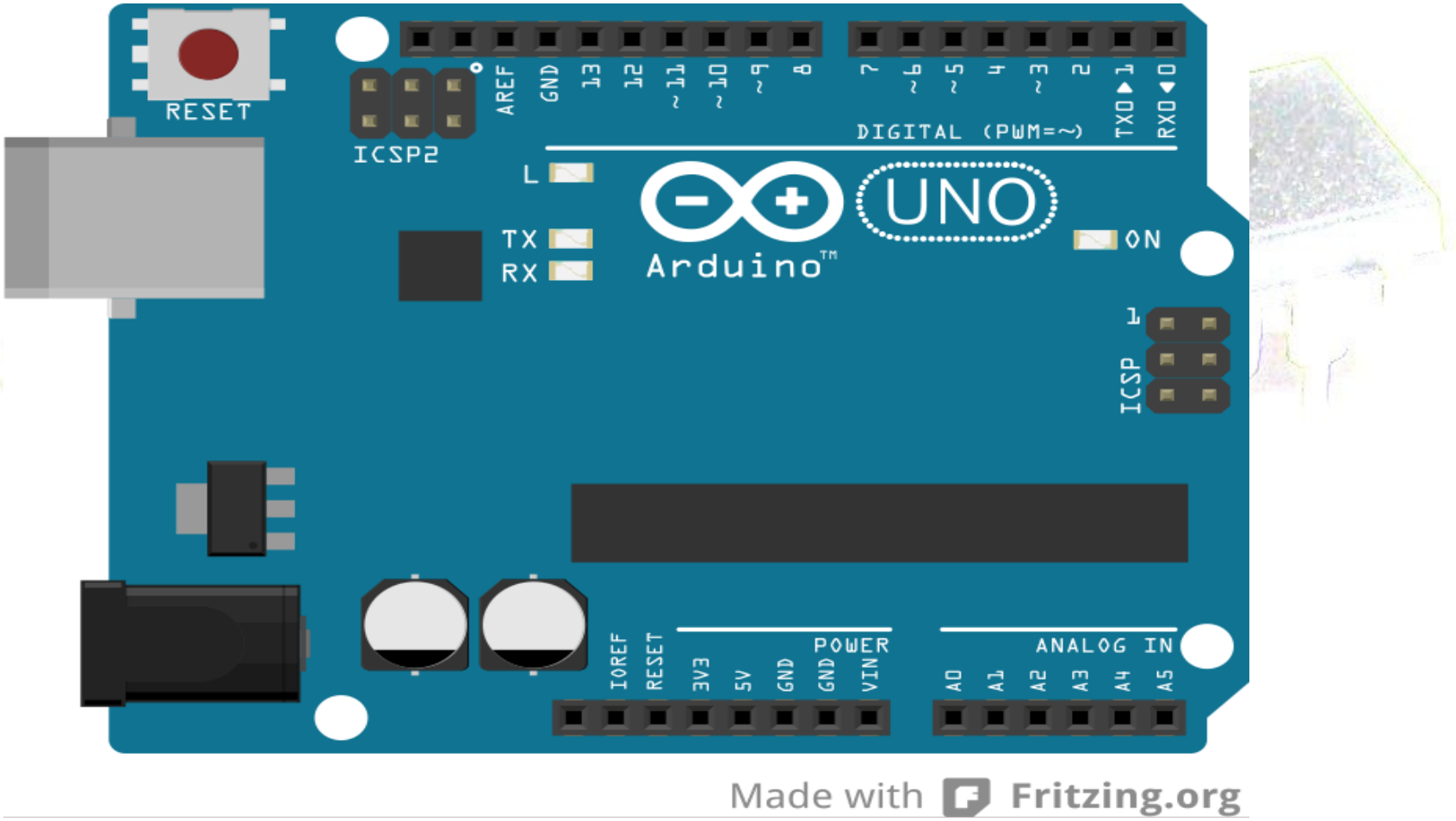
Dispositivo interattivo



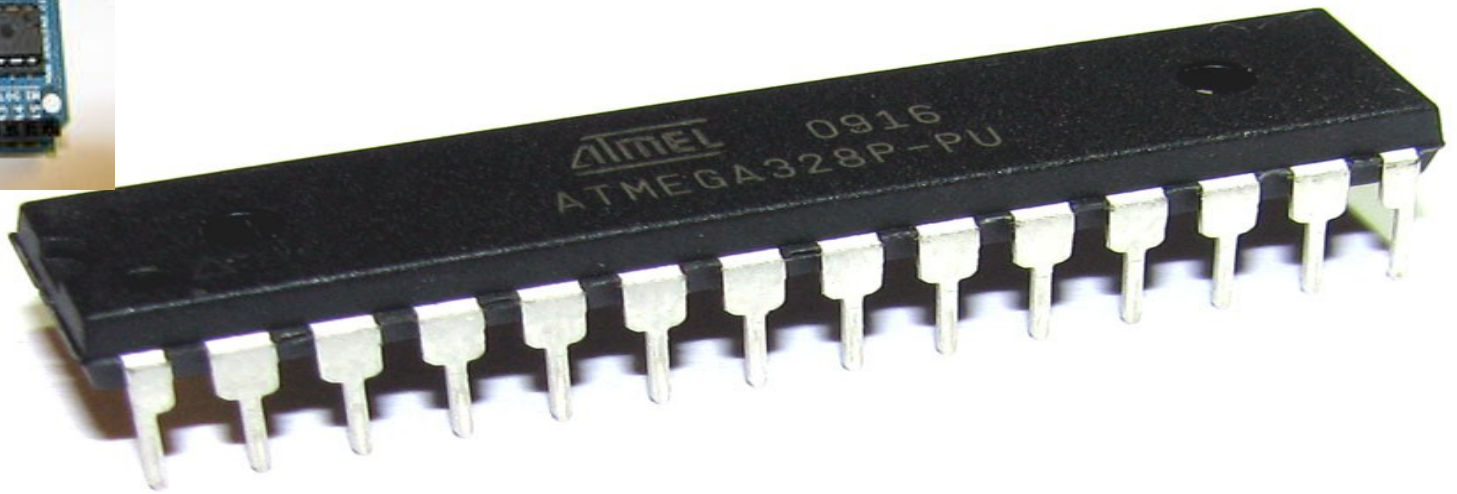
Il progetto Arduino è nato nel 2005 in Italia presso l'*Interaction Design Institute di Ivrea* con l'intento di fornire agli studenti uno strumento semplice ed economico di prototipazione elettronica.

La scheda e il software di sviluppo sono nati con licenza open source . Arduino è un progetto in continua evoluzione.

Arduino UNO

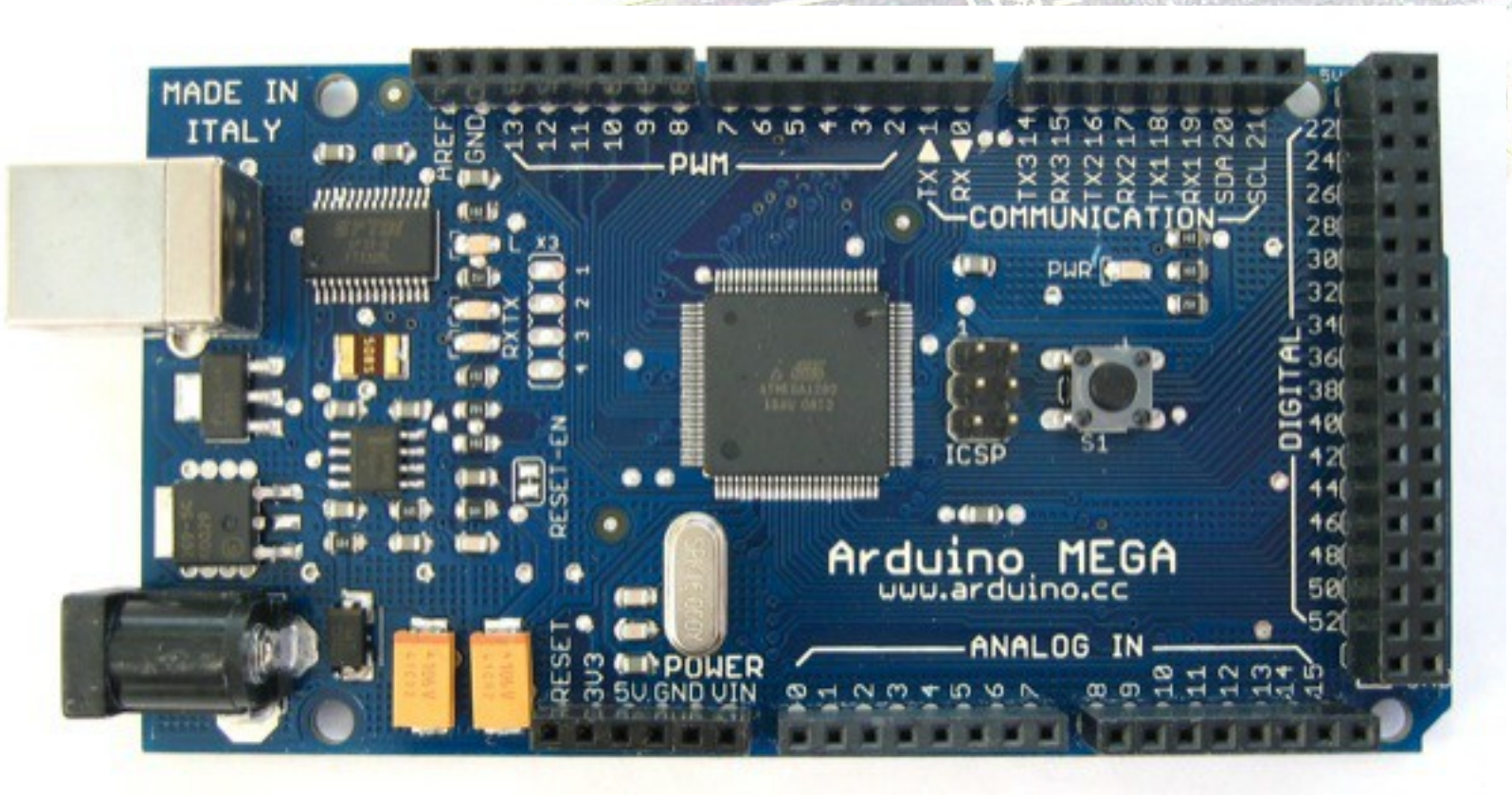

RESET
ICSP2
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM=~)
L
TX
RX
Arduino™
UNO
ON
ICSP
IOREF
RESET
3V3
5V
GND
GND
VIN
POWER
A0
A1
A2
A3
A4
A5
ANALOG IN
Made with Fritzing.org

Il microcontrollore utilizzato è una
ATMEGA 328 a 16 MHz
Memoria flash 32 Kbyte
Memoria sram 2 Kbyte
32 registri a 8 bit uso generale



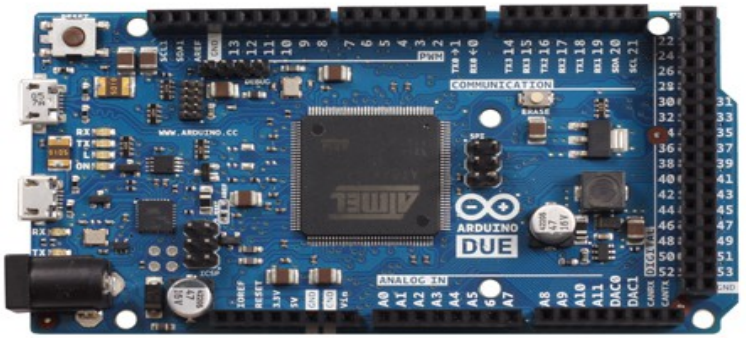
ALTRE SCHEDE ARDUINO

ARDUINO MEGA ATMEGA 2560 16 Mhz
54 I/O digitali 16 analogici



ALTRE SCHEDE ARDUINO

ARDUINO LEONARDO ATMEGA 32U4 16 Mhz
USB e seriale separati



ARDUINO DUE 32 bit ARM 84 MHz

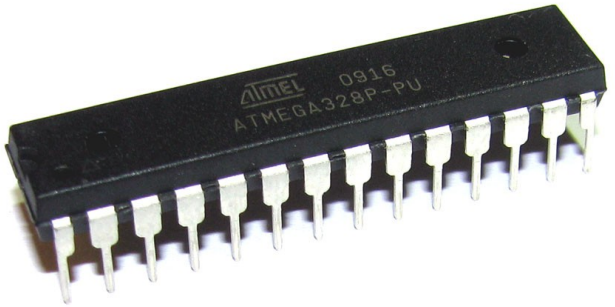
ARDUINO YUN ATMEGA 32U4 come Ard. LEONARDO
16 Mhz **+** ATHEROS 9331 400 Mhz con LINUX e WiFi

Il microcontrollore utilizzato è una ATMEGA 328 a 16 MHz Memoria flash 32 KByte

Nel processore è memorizzato un piccolo programma di boot che serve a caricare nella memoria flash del microcontrollore lo sketch creato con l'ambiente di sviluppo su Pc, IDE.

Il processore è di tipo RISC e una istruzione macchina viene eseguita mediamente in un solo ciclo di clock.

L'architettura è di tipo Harvard con canali di comunicazione dati e istruzioni separati.



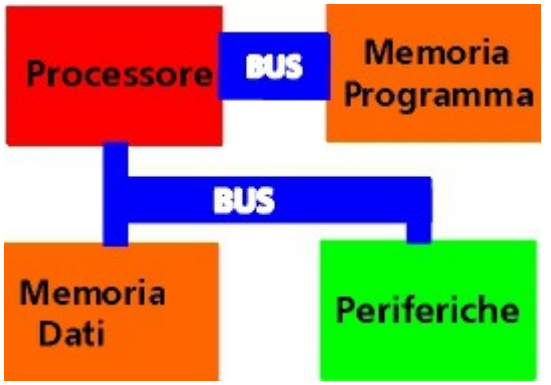
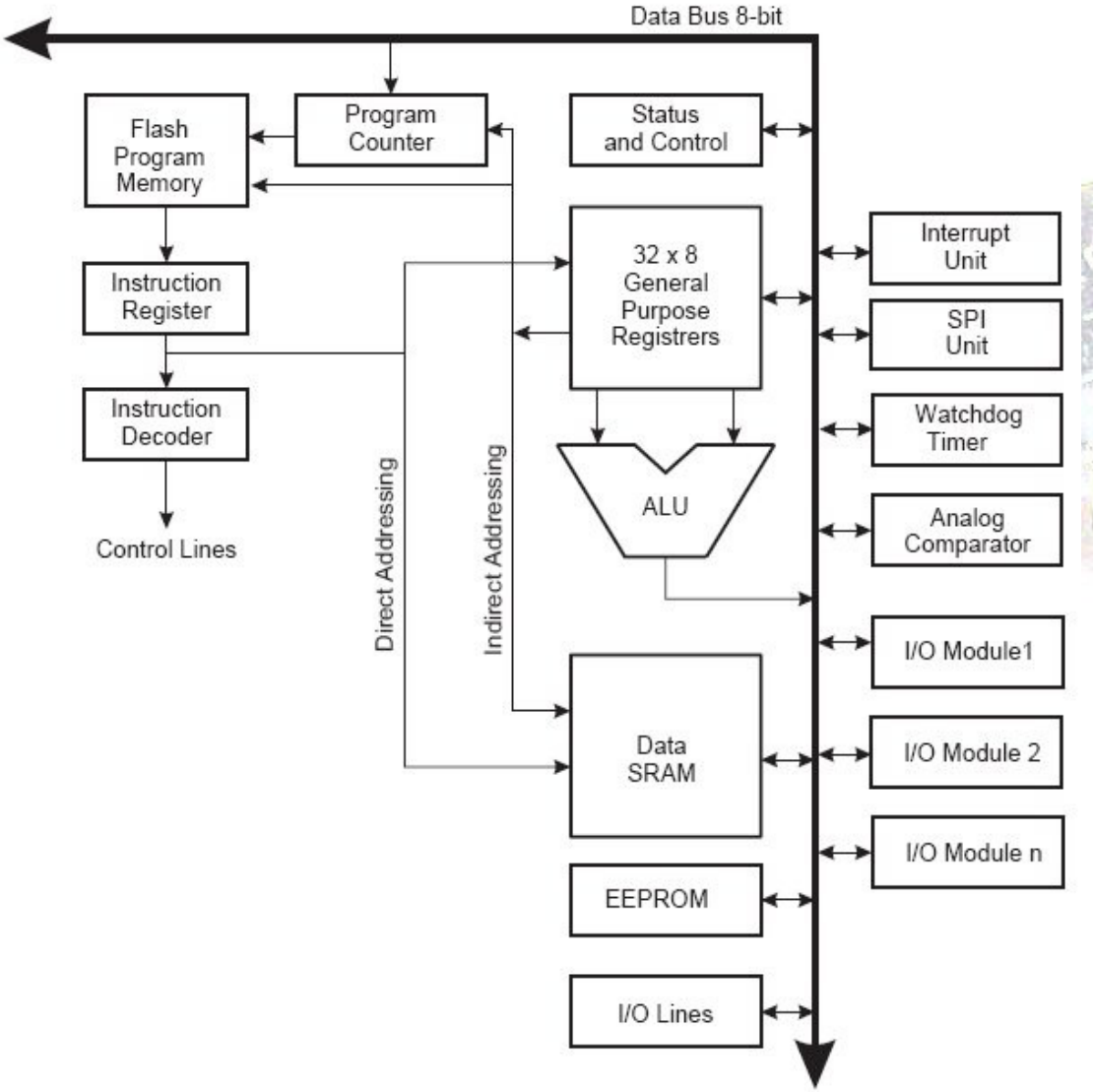
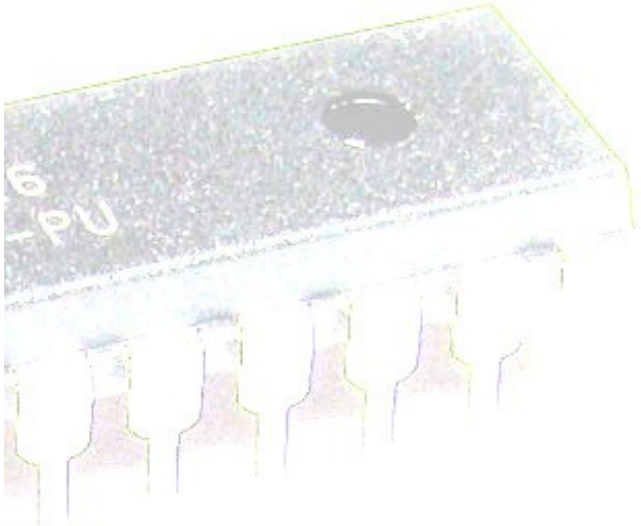
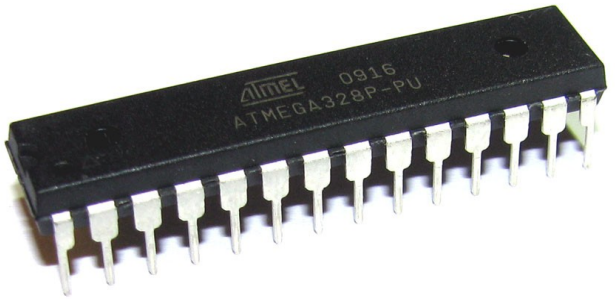


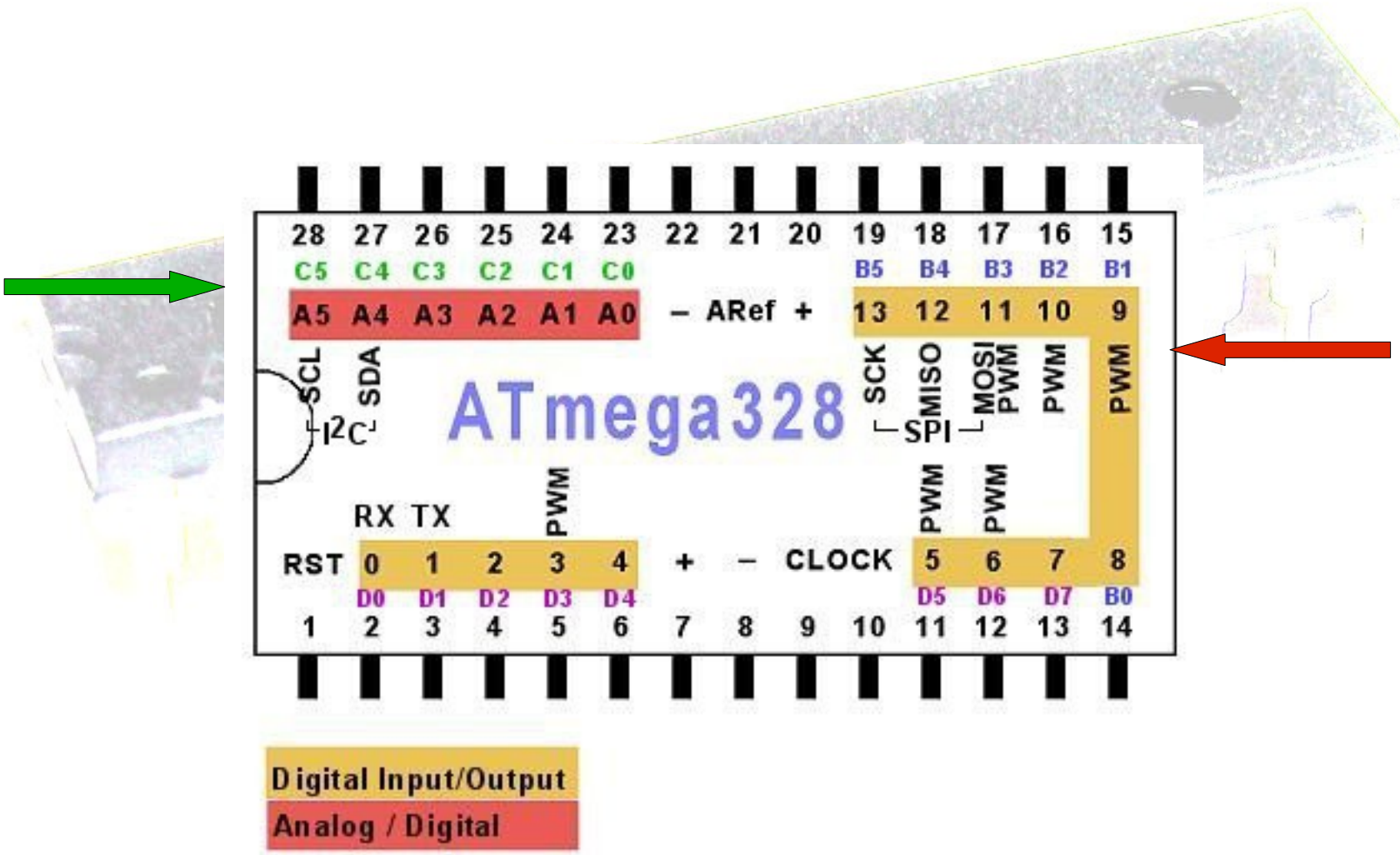
Figure 6-1. Block Diagram of the AVR Architecture



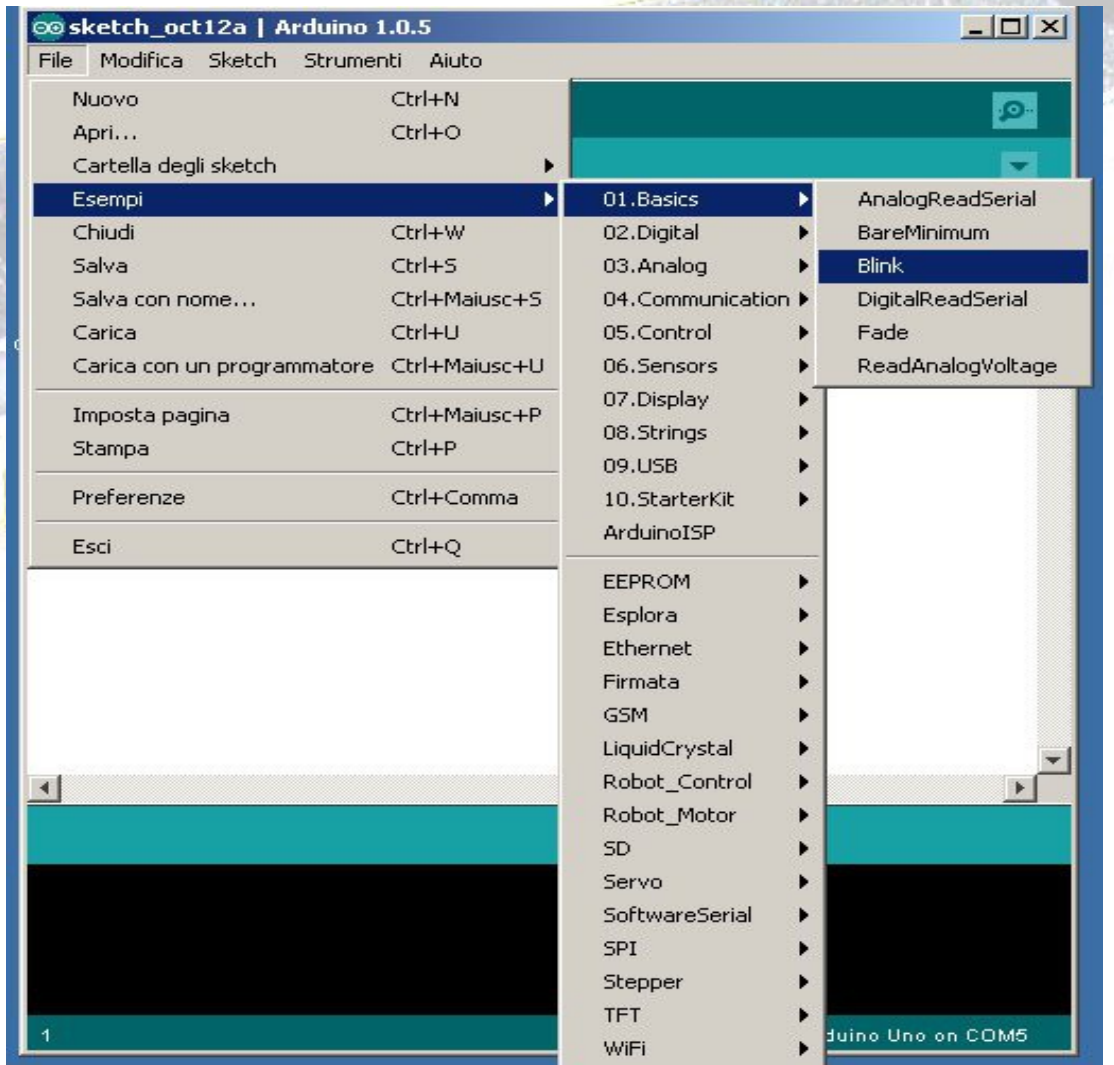




6 ingressi analogici
14 I/O digitali (6 pwm)



Con l'ambiente di sviluppo IDE si scrivono i programmi chiamati sketch (schizzo)
Come visualizzare uno sketch di esempio denominato blink

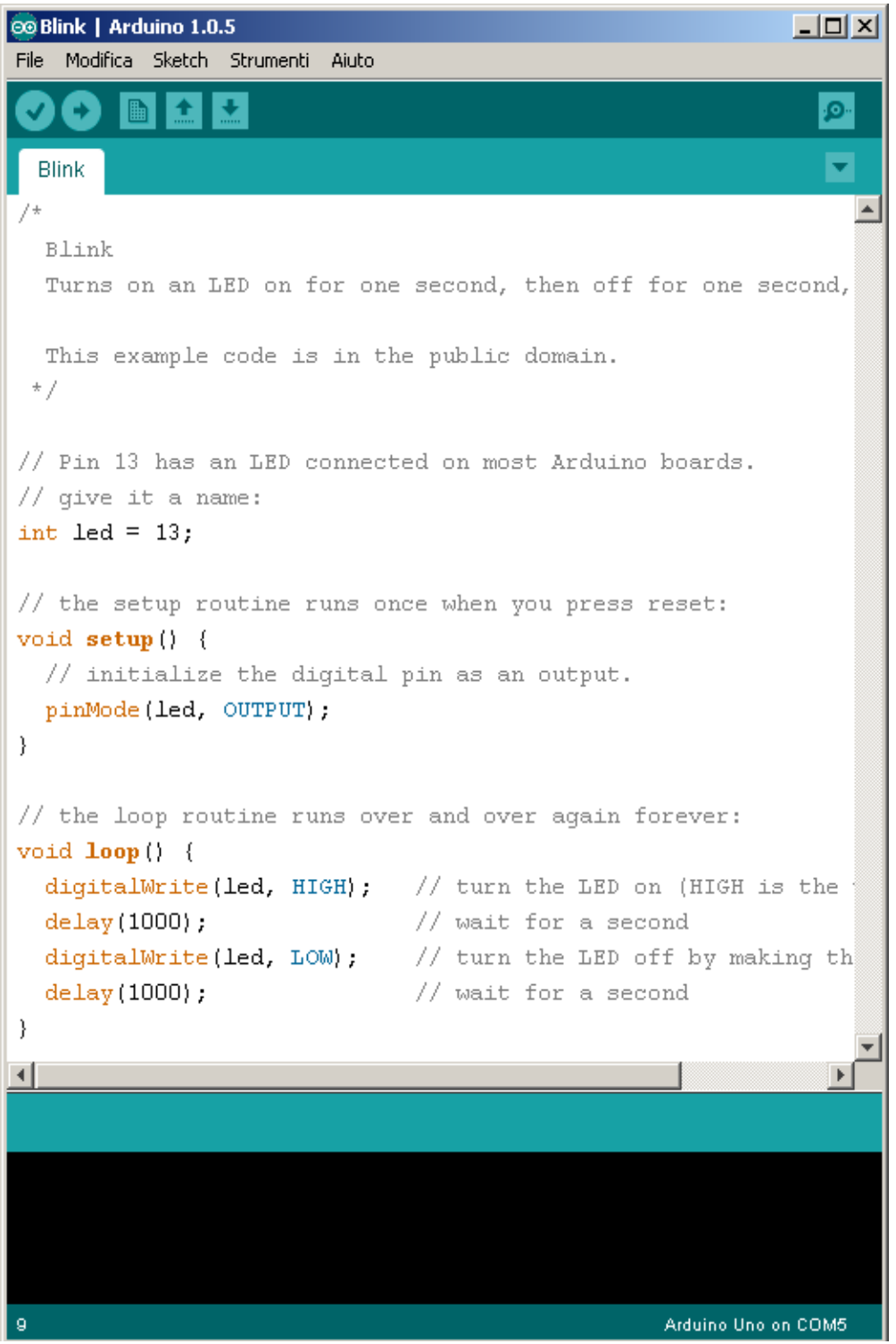


Ci sono due funzioni: **setup** che viene eseguita una sola volta e **loop** che è un ciclo infinito.

pinMode(pin,OUTPUT-INPUT)

digitalWrite(pin,HIGH-LOW)

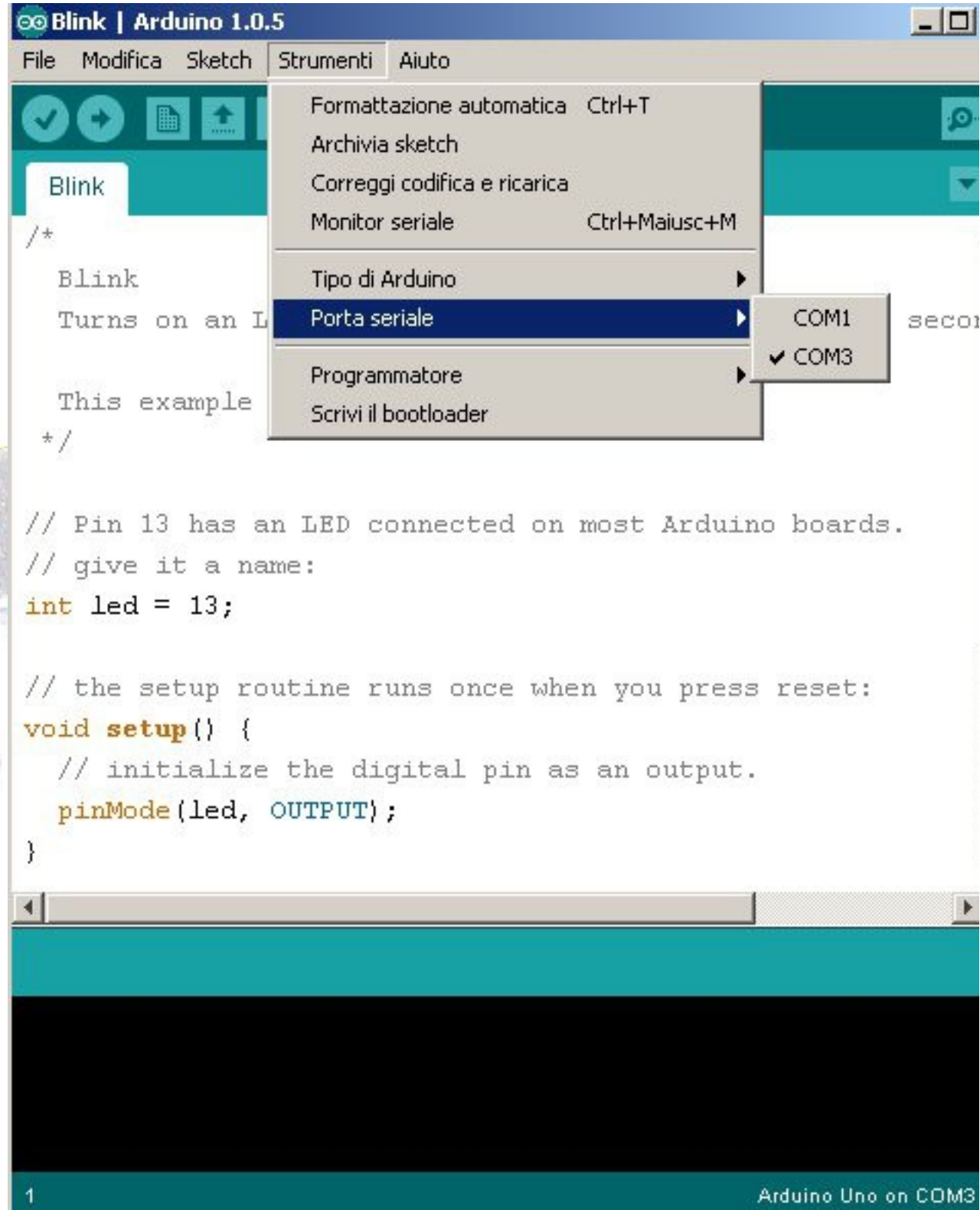
delay(millisec)



Avviato l'IDE e collegata la scheda di Arduino è necessario selezionare il tipo di scheda e la connessione seriale da utilizzare. Quando la scheda Arduino viene connessa al pc nasce una porta virtuale seriale.

La connessione seriale può essere utilizzata anche da programma per una comunicazione tra Arduino e Pc.

Serial.begin(9600);
Serial.print(dato);
Serial.read();



Il linguaggio usato è un C C++ con alcune funzioni specifiche.

Alcune funzioni

```
pinMode(pin, mode);   // imposta il pin in INPUT o OUTPUT
digitalRead(pin);         // legge un ingresso digitale
digitalWrite(pin, value)    // invia HIGH o LOW al pin di uscita
analogRead(pin);       // legge da un ingresso analogico  0-1023
delay(millisec)         // ritardo
```

Costanti e variabili

Le variabili appartengono alle locazioni di memoria RAM che al cessare dell'alimentazione perdono il loro contenuto.

boolean true false

char un carattere ASCII. Un byte in memoria

byte memorizza un intero da 0 a 255

int numero intero da -32768 a 32767 . Due byte

unsigned int numero da 0 a 65535

long memorizza un numero tra -2147483648 e 2147483647

float numero in virgola mobile, 7 cifre dopo il punto decimale.

In memoria 4 byte

double virgola mobile doppia precisione

Costanti e variabili

Per creare un array:

int nome_variabile[3] ; crea tre variabili adiacenti di tipo intero

char nome_variabile[8] = “ Arduino”; crea una stringa di 7 caratteri più un carattere terminatore. 8 byte da 0 a 7

Modo veloce per definire il contenuto di un array:

int nome_variabile[]= {0, 25, 50, 100, 200};

Le variabili nel linguaggio di programmazione C++ che usa Arduino hanno una proprietà di visibilità denominata scope.

Una variabile globale è visibile da ogni funzione del programma.

Le variabili locali sono visibili soltanto all'interno della funzione nella quale esse sono dichiarate.

Nell'ambiente Arduino qualsiasi variabile dichiarata fuori di una funzione (per es. setup(), loop(), etc.) è una variabile globale.

Le variabili locali sono un modo utile per assicurare che soltanto all'interno di quella funzione si ha accesso alle proprie variabili. Questo previene errori di programmazione quando una funzione inavvertitamente modifica variabili usate da un'altra funzione.

E' anche comodo dichiarare e inizializzare una variabile all'interno di un ciclo. Questo crea una variabile accessibile solo all'interno di un ciclo for.

```
for( int k=0;k<10;k++) {
...
```

Funzioni e parametri

All'interno di una funzione sono visibili le variabili globali
I passaggio dei parametri può avvenire per valore o per riferimento

globali

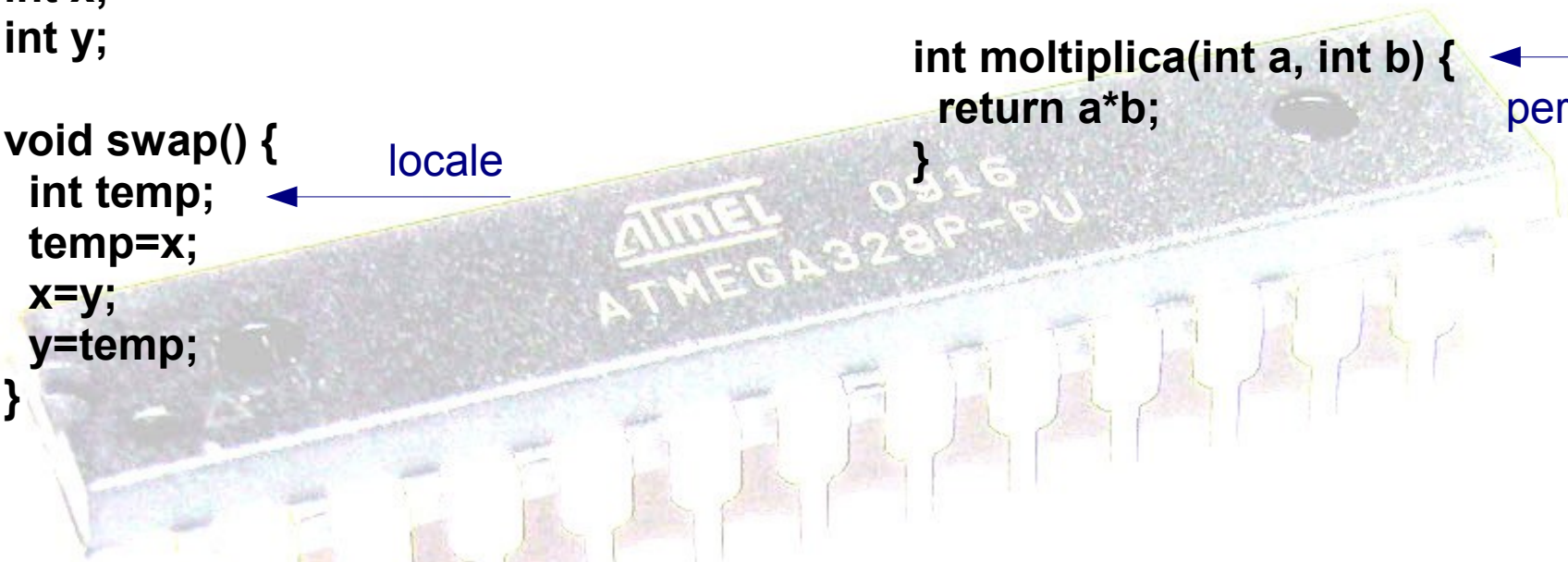
```
int x;
int y;
```

```
void swap() {
  int temp;
  temp=x;
  x=y;
  y=temp;
}
```

locale

```
int moltiplica(int a, int b) {
  return a*b;
}
```

per valore




```
void swapr(int &val1, int &val2) {
  int temp;
  temp=val1;
  val1=val2;
  val2=temp;
/* in questo caso la funzione
modifica due valori
utilizzo:    swapr(x,y)*/
}
```

per riferimento

Istruzioni di controllo

if else

```
if (variabile == 55) {
    digitalWrite(3,1);
}
else
   digitalWrite(3,0);
```

for

```
for(int i=0; i< 10; i++) {
   Serial.println("Salve");
 }
```

while

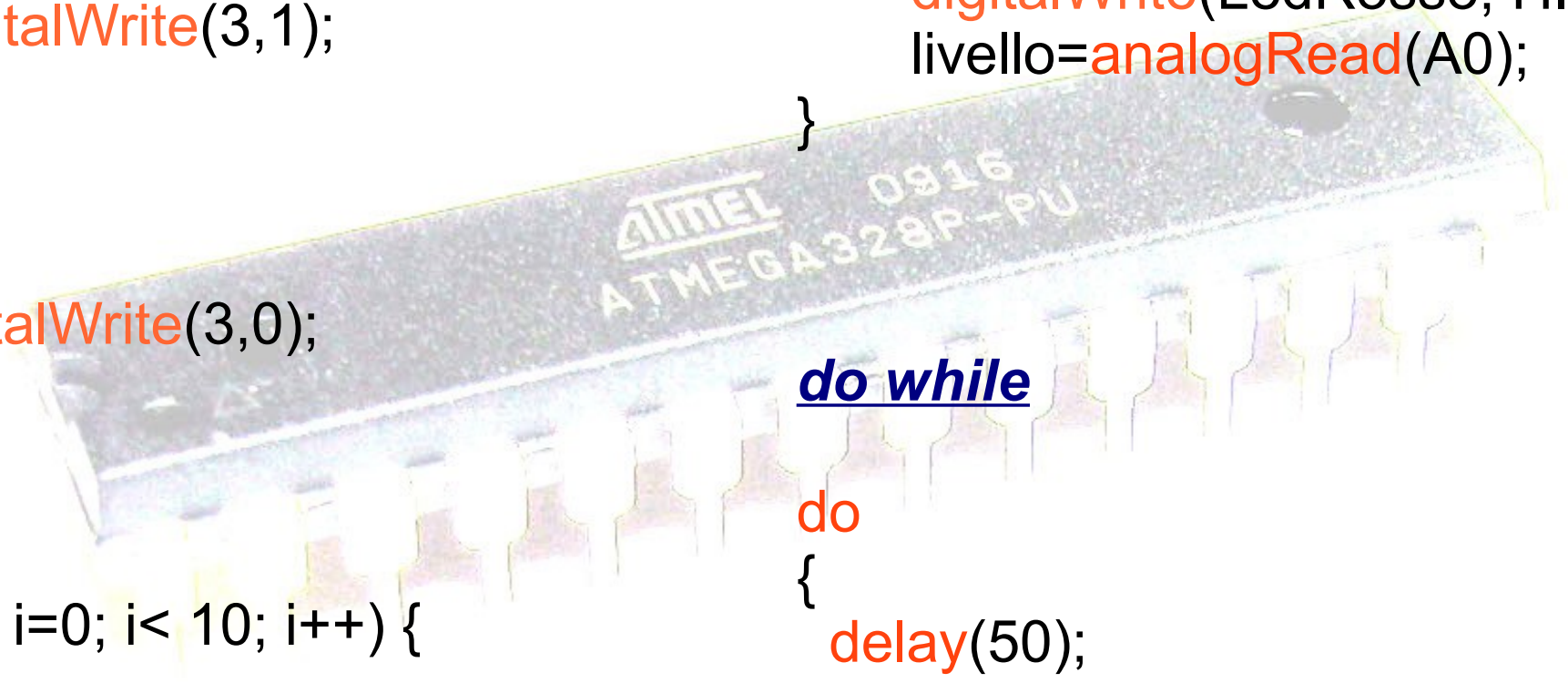
condizione

```
while(livello < 512) {
      digitalWrite(LedRosso, HIGH);
      livello=analogRead(A0);
}
```

do while

```
do
{
  delay(50);
  x = analogRead(A1);  //

} while (x < 100);
```

condizione



Istruzioni di controllo

break

break è usata per uscire da do, for, o while , bypassando la condizione normale del ciclo.

```
while (true) {
  if(digitalRead(5)==0)
        break;
 }
```

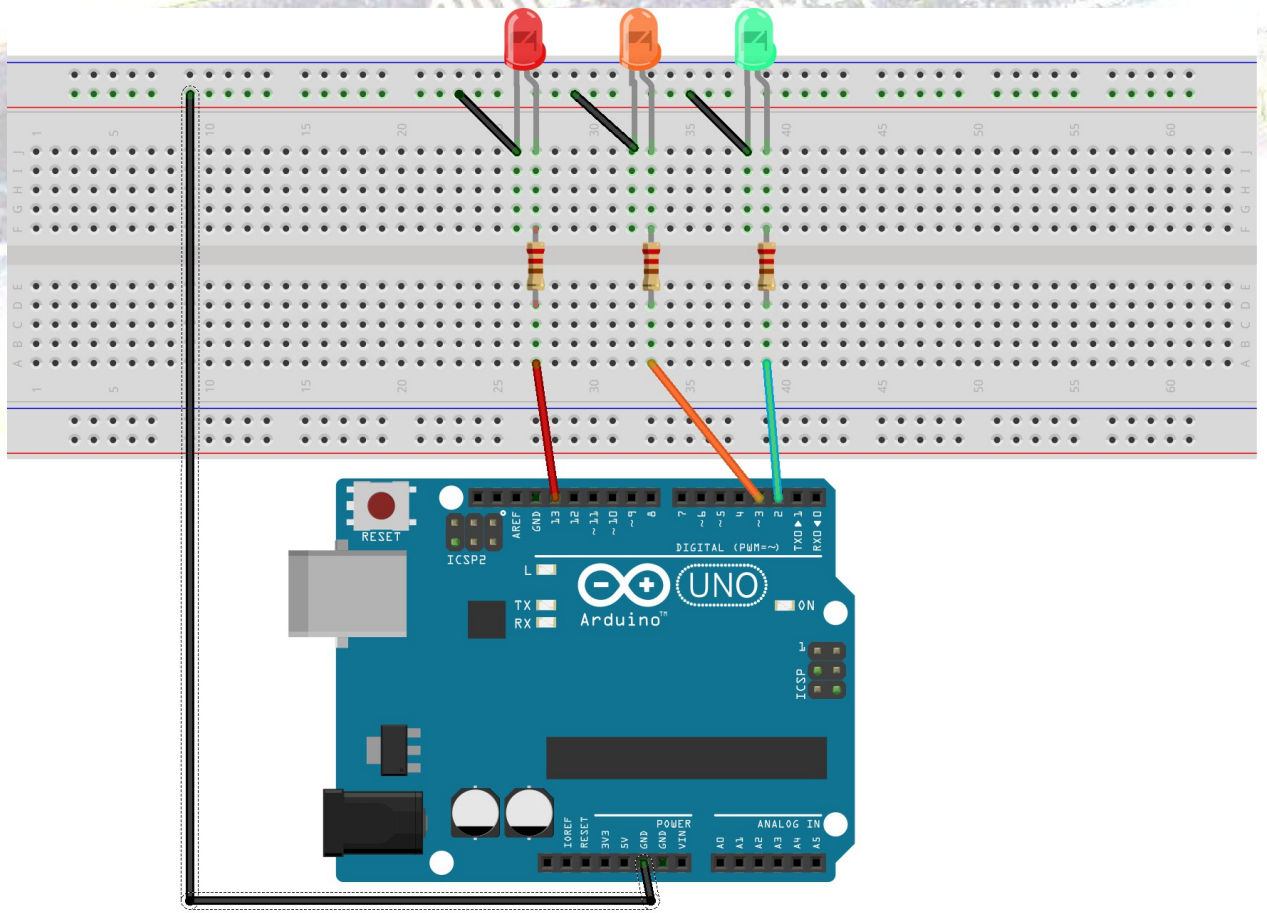
switch case

```
switch (var) {
    case 1:
      //esegui  quando var==1
      break;
    case 2:
      //esegui quando  var==2
      break;
    default:
      // esegui il default
      // default è opzionale
 }
```

Accendere e spegnere led

Problema

Si vuole che il proprio sketch accenda e spenga tre diodi led in modo da simulare il funzionamento di un semaforo.

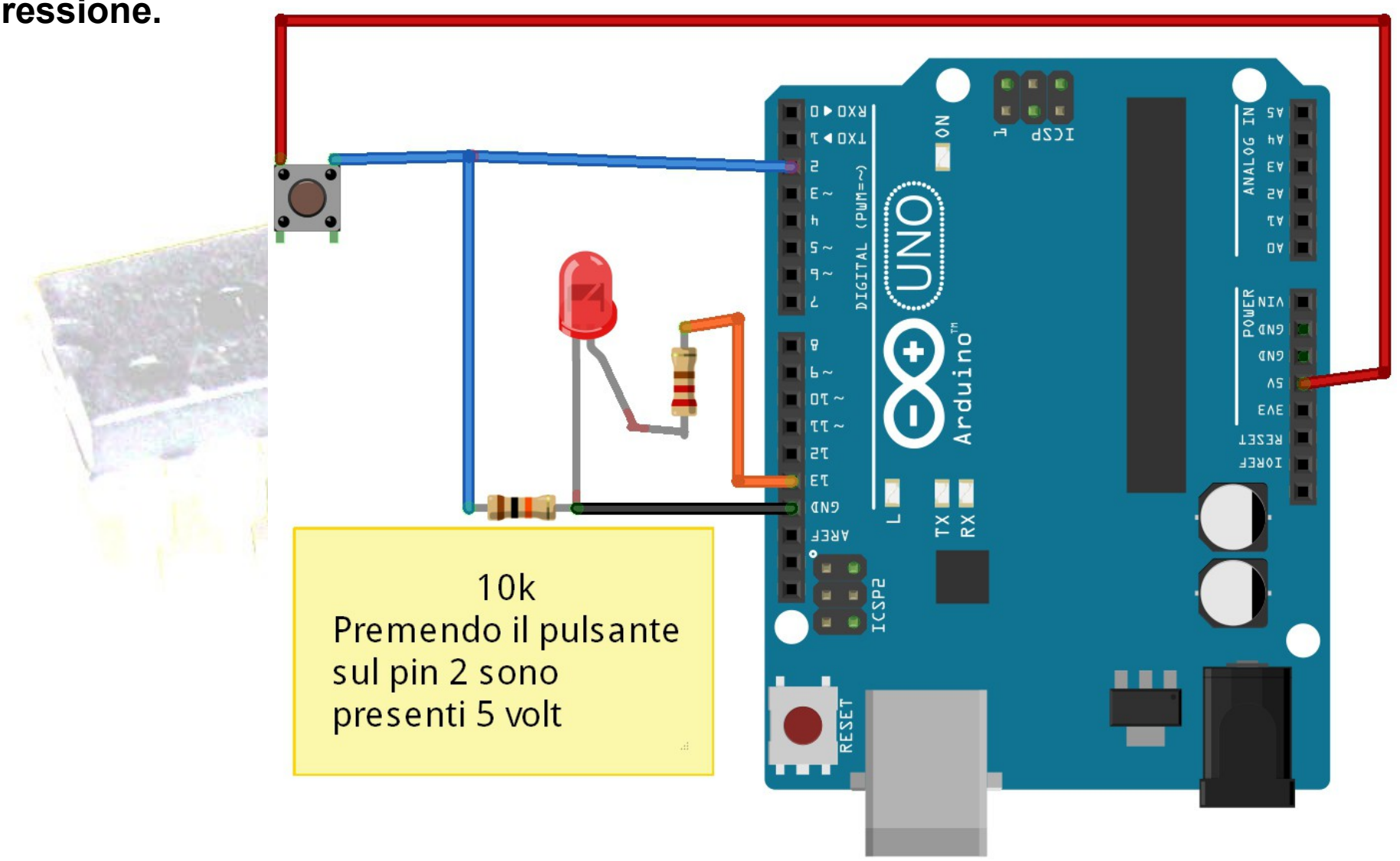



semaforo_oct25a | Arduino 1.0.5
File Modifica Sketch Strumenti Aiuto
semaforo_oct25a
const int verde=2;
const int giallo=3;
const int rosso=13;
void setup()
{
pinMode(verde,OUTPUT);
pinMode(giallo,OUTPUT);
pinMode(rosso,OUTPUT);
}
void loop(){
digitalWrite(rosso,LOW);
digitalWrite(verde,HIGH);
delay(4000);
digitalWrite(giallo,HIGH);
digitalWrite(verde,LOW);
delay(1000);
digitalWrite(giallo,LOW);
digitalWrite(rosso,HIGH);
delay(4000);
}
Salvataggio effettuato.
Dimensione del file binario dello sketch: 1.136 bytes (su un massimo di 32.256 bytes)

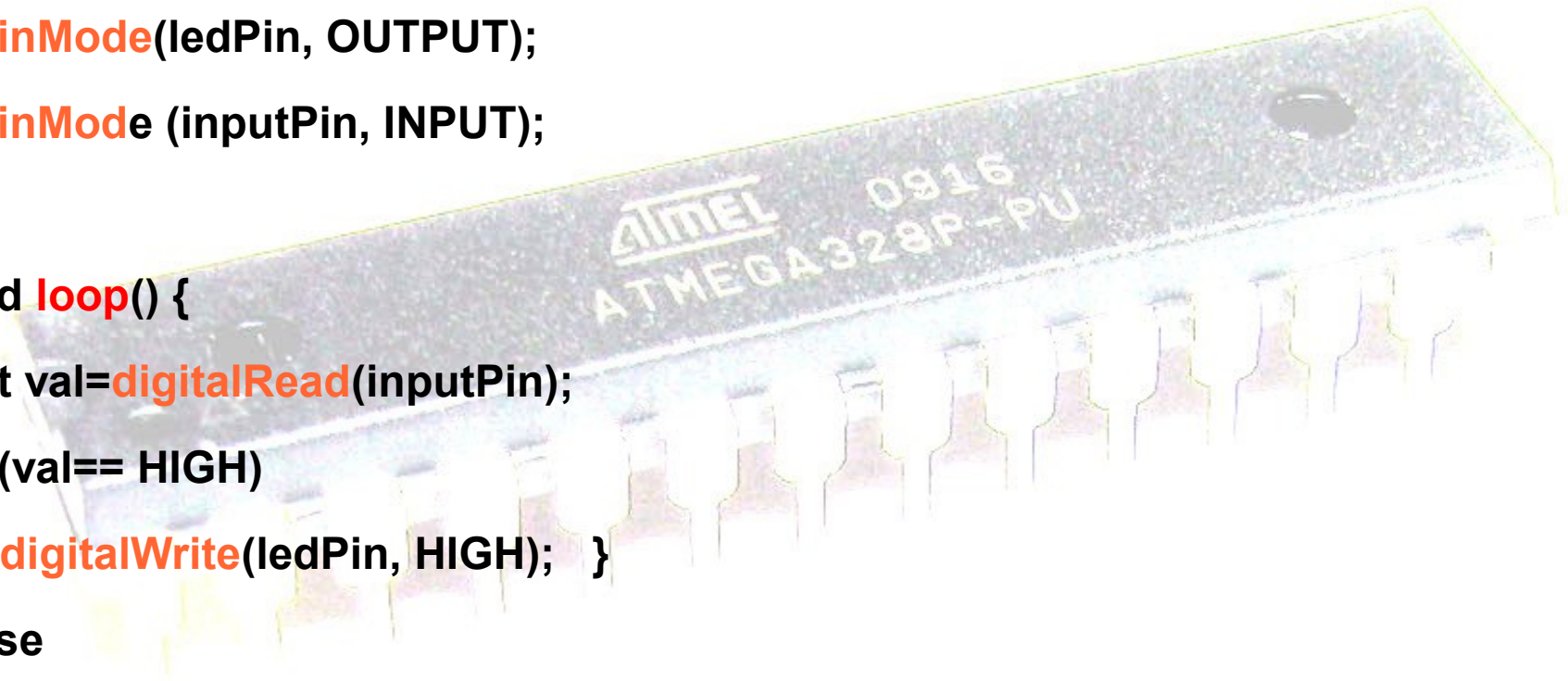
Utilizzare un pulsante

Problema

Si vuole che il proprio sketch risponda accendendo un diodo led quando viene chiuso un contatto elettrico, per esempio quando si utilizza un pulsante a pressione.



```
/* un pulsante collegato al pin 2 accende un led collegato al pin 13 */
const int ledPin = 13;
const int inputPin = 2;
void setup(){
   pinMode(ledPin, OUTPUT);
   pinMode (inputPin, INPUT);
}
void loop() {
  int val=digitalRead(inputPin);
  if (val== HIGH)
  {  digitalWrite(ledPin, HIGH);   }
  else
  { digitalWrite(ledPin,LOW);
  }
}
```



Riconoscere la pressione del pulsante

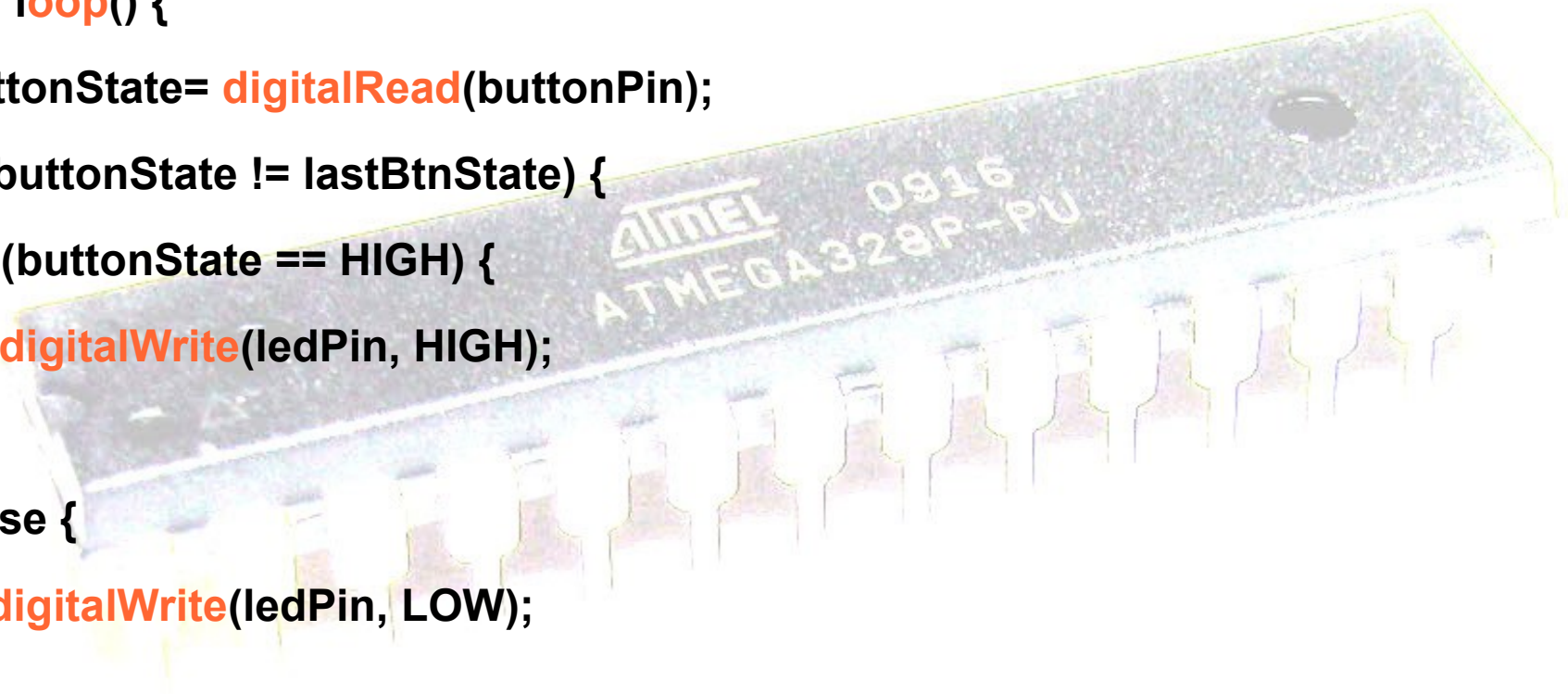
Problema

Si vuole che lo sketch agisca su un led in uscita solo in corrispondenza del cambiamento di stato del pulsante.

Soluzione

Si utilizza una variabile per memorizzare lo stato e agire solo quando questo valore cambia.

```
int buttonState = 0;
int lastBtnState = LOW; // stato precedente del pulsante


void loop() {
  buttonState= digitalRead(buttonPin);
  if (buttonState != lastBtnState) {
    if (buttonState == HIGH) {
      digitalWrite(ledPin, HIGH);
    }
    else {
      digitalWrite(ledPin, LOW);
    }
  }
lastBtnState= buttonState;
}
```



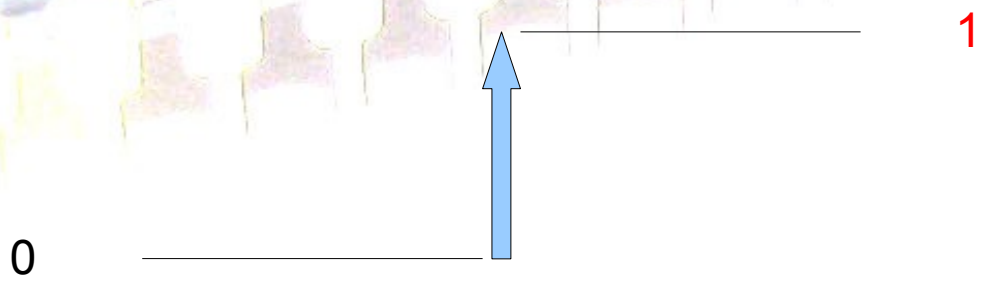
Riconoscere la pressione del pulsante per cambiare lo stato del led in uscita.

Problema

Si vuole che lo sketch agisca in uscita in modo che il led si illumini ad una prima pressione del tasto e resti acceso spegnendosi alla pressione successiva.

Soluzione

Lo sketch precedente, grazie alle funzioni if nidificate, è in grado di distinguere la pressione (*rising edge*) dal rilascio (*falling edge*) del pulsante. Per fare in modo che il led resti acceso al rilascio del pulsante reagiremo solo sul *rising edge*.



```
const int buttonPin = 2;const
int ledPin = 13;


int buttonState = 0;

int lastBtnState = LOW;

int ledState= LOW;


void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(buttonPin, INPUT);
}
```

```
void loop() {
  buttonState= digitalRead(buttonPin);
  if (buttonState != lastBtnState &&
buttonState == HIGH) {
    ledState = ! ledState;
    if (ledState== HIGH) {
     digitalWrite(ledPin, HIGH);
    }
    else {
      digitalWrite(ledPin, LOW);
    }
  }
 lastBtnState= buttonState;
}
```

Una complicazione: il debounce di un pulsante.

Problema

Realizzando lo sketch precedente ci siamo imbattuti in una complicazione non banale, il led si accende o si spegne quando non dovrebbe. Il problema è dovuto ai falsi contatti del pulsante durante la sua pressione o al suo rilascio, fenomeno che ha una durata di qualche decina di millisecondi.

Soluzione

Usare tecniche di debaunce, la più semplice consiste nell'aggiungere un ritardo dopo l'esecuzione del codice legato alla pressione del tasto.

Schema semplificato del fenomeno

```
const int buttonPin = 2;

const int ledPin = 13;

const int debaunceDelay = 50;


int buttonState = 0;

int lastBtnState = LOW;

int ledState= LOW;


void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(buttonPin, INPUT);
}
```

```
void loop() {
  buttonState= digitalRead(buttonPin);
  if (buttonState != lastBtnState &&
buttonState == HIGH) {
    ledState = ! ledState;
    if (ledState== HIGH) {
     digitalWrite(ledPin, HIGH);
    }
    else {
      digitalWrite(ledPin, LOW);
    }
  }
 lastBtnState= buttonState;
 delay(debaunceDelay);
}
```

Simulazione semaforo con due funzioni normale e lampeggiante

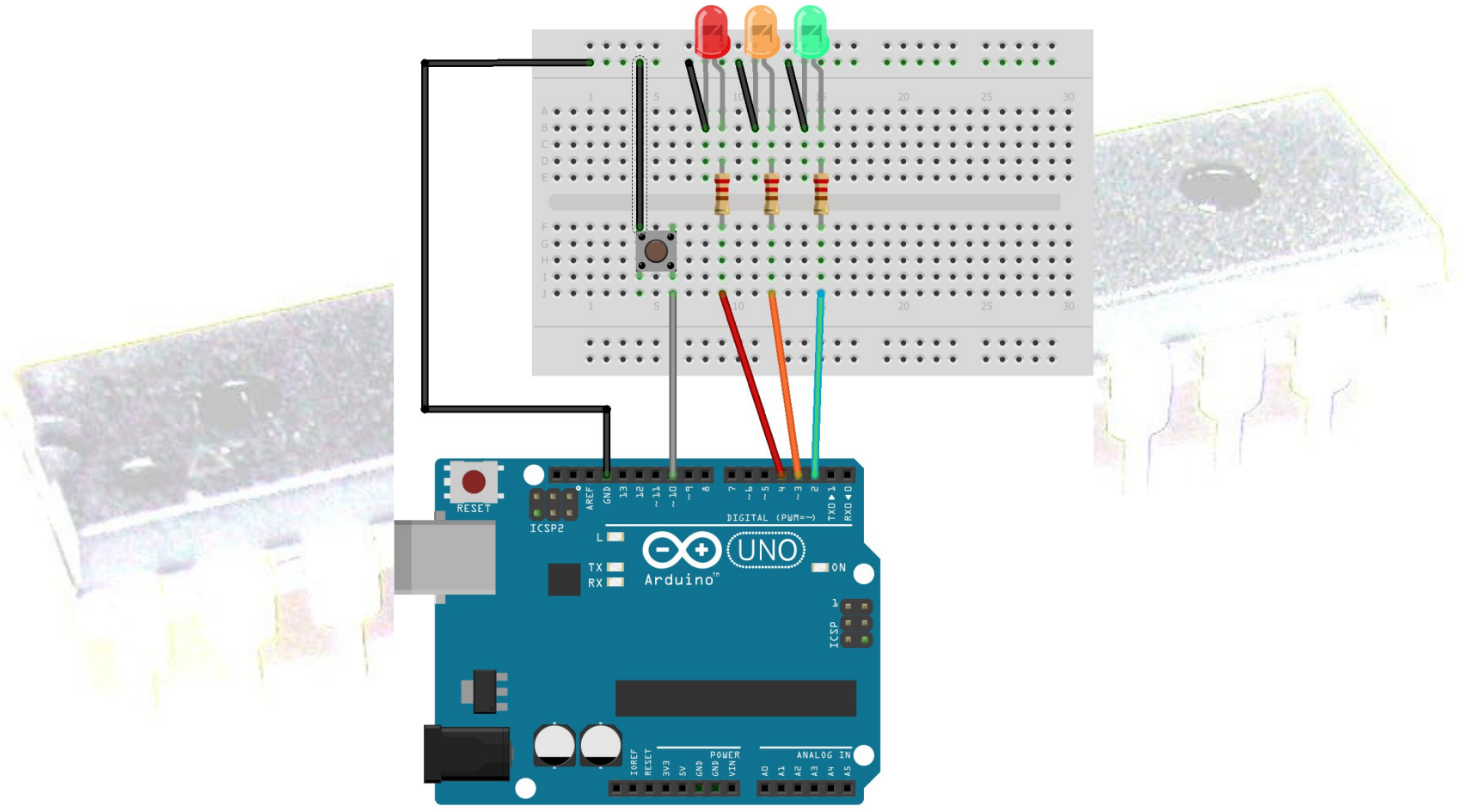
Problema

Si vuole che il semaforo realizzato con tre led passi dal funzionamento normale a quello con giallo lampeggiante premendo un pulsante e viceversa.

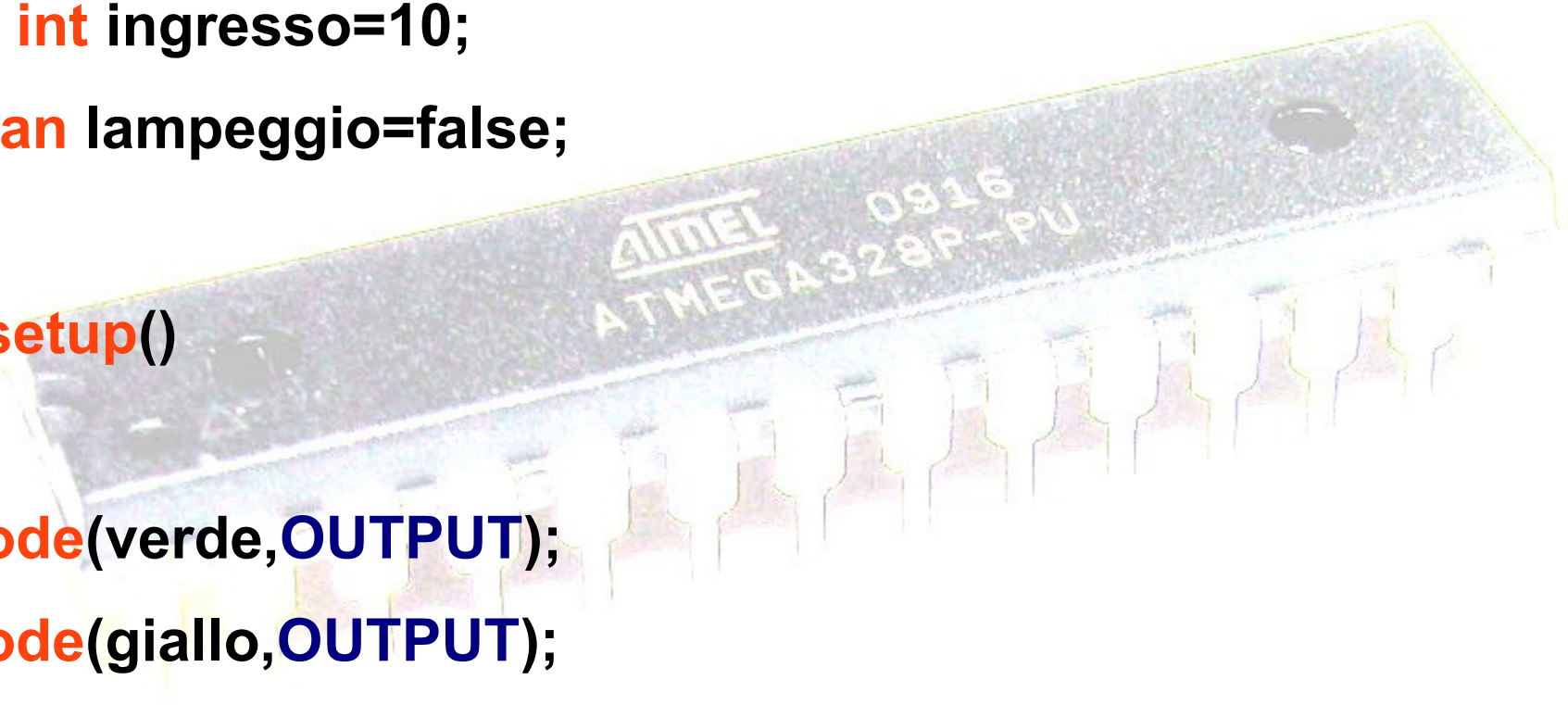
Soluzione

E' opportuno utilizzare due funzioni una per il funzionamento normale e l'altra per quello a lampeggio. Una variabile di stato determina quale delle due utilizzare.

Il pulsante premuto determina il cambiamento della variabile di stato.

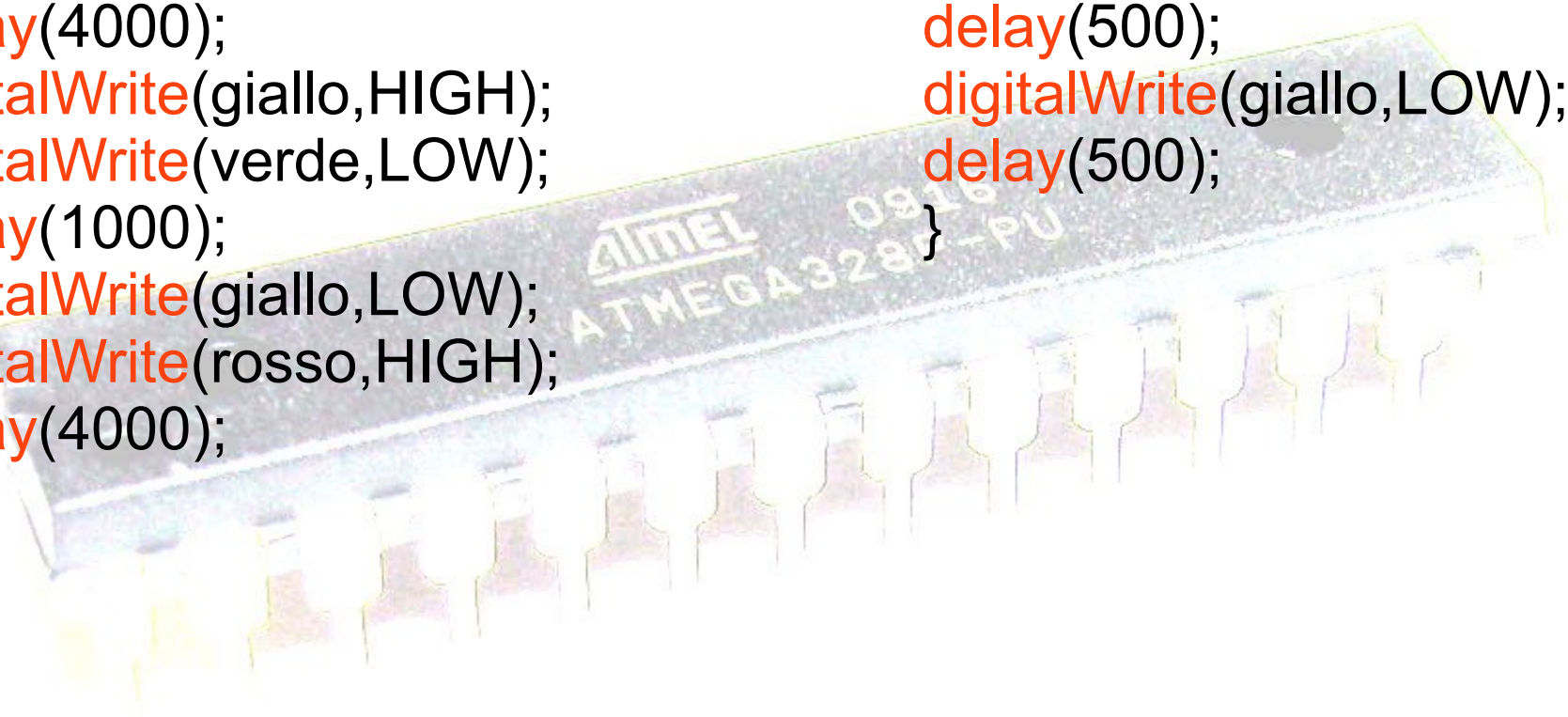

RESET
ICSP2
AREF
GND
13
12
~ 11
~ 10
~ 9
8
7
~ 6
~ 5
4
~ 3
2
TXD → 1
RXD ← 0
DIGITAL (PWM=~)
L
TX
RX
Arduino™
UNO
ON
ICSP
POWER
IOREF
RESET
3V3
5V
GND
GND
VIN
ANALOG IN
A0
A1
A2
A3
A4
A5
Made with Fritzing.org

```
const int verde=2;
const int giallo=3;
const int rosso=4;
const int ingresso=10;
boolean lampeggio=false;

void setup()
{
pinMode(verde,OUTPUT);
pinMode(giallo,OUTPUT);
pinMode(rosso,OUTPUT);
pinMode(ingresso,INPUT);
digitalWrite(ingresso,HIGH); // attivazione pull up
}          // in alternativa pinMode(ingresso, INPUT_PULLUP);
```

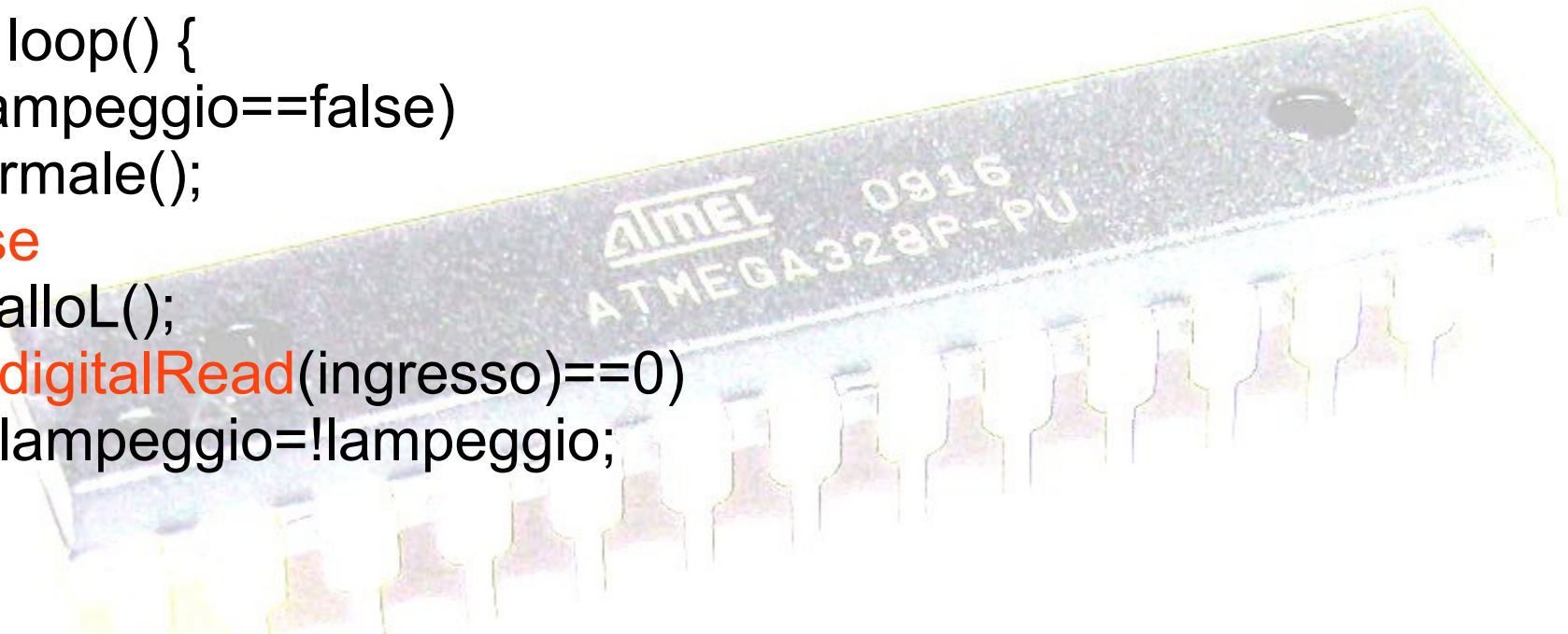


```
void normale(){
  digitalWrite(rosso,LOW);
  digitalWrite(verde,HIGH);
  digitalWrite(giallo,LOW);
  delay(4000);
  digitalWrite(giallo,HIGH);
  digitalWrite(verde,LOW);
  delay(1000);
  digitalWrite(giallo,LOW);
  digitalWrite(rosso,HIGH);
  delay(4000);
  }
```

```
void gialloL() {
    digitalWrite(verde,LOW);
    digitalWrite(rosso,LOW);
    digitalWrite(giallo,HIGH);
    delay(500);
    digitalWrite(giallo,LOW);
    delay(500);
    }
```

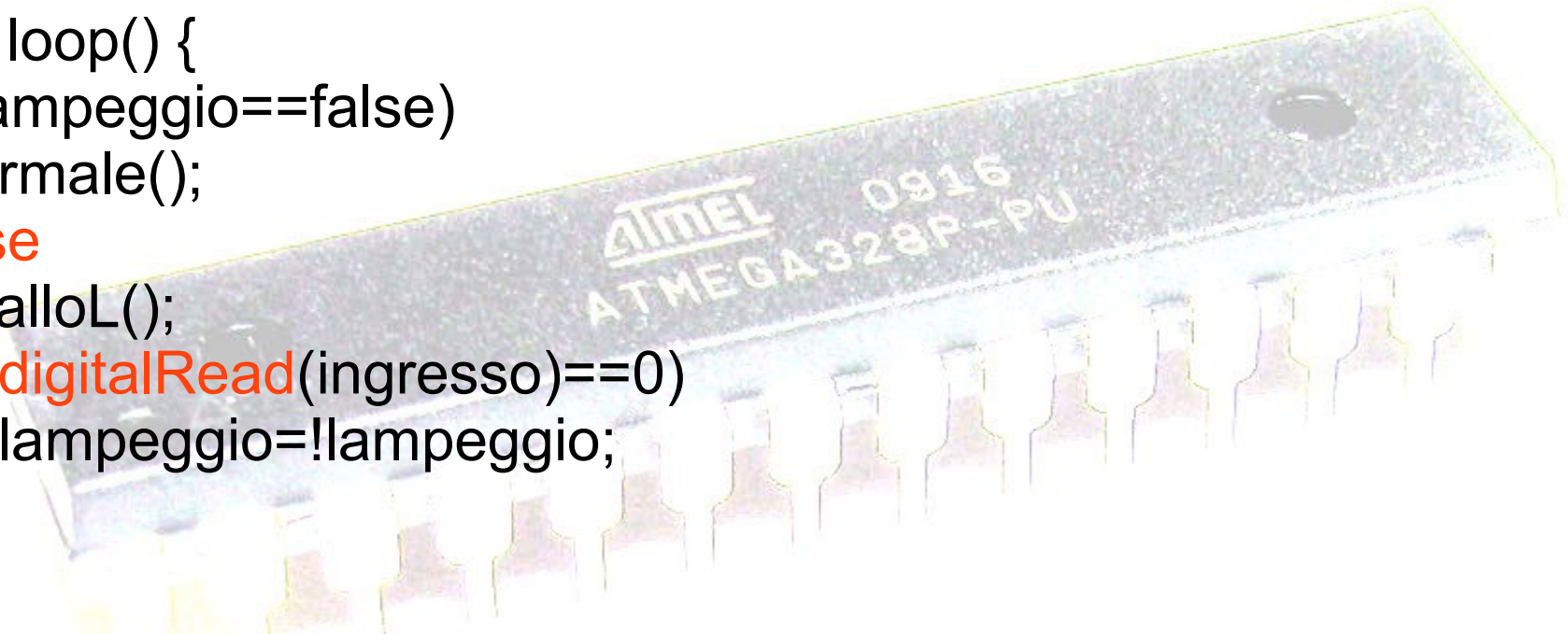


```
void loop() {
  if(lampeggio==false)
   normale();
   else
    gialloL();
   if (digitalRead(ingresso)==0)
       lampeggio=!lampeggio;
}
```



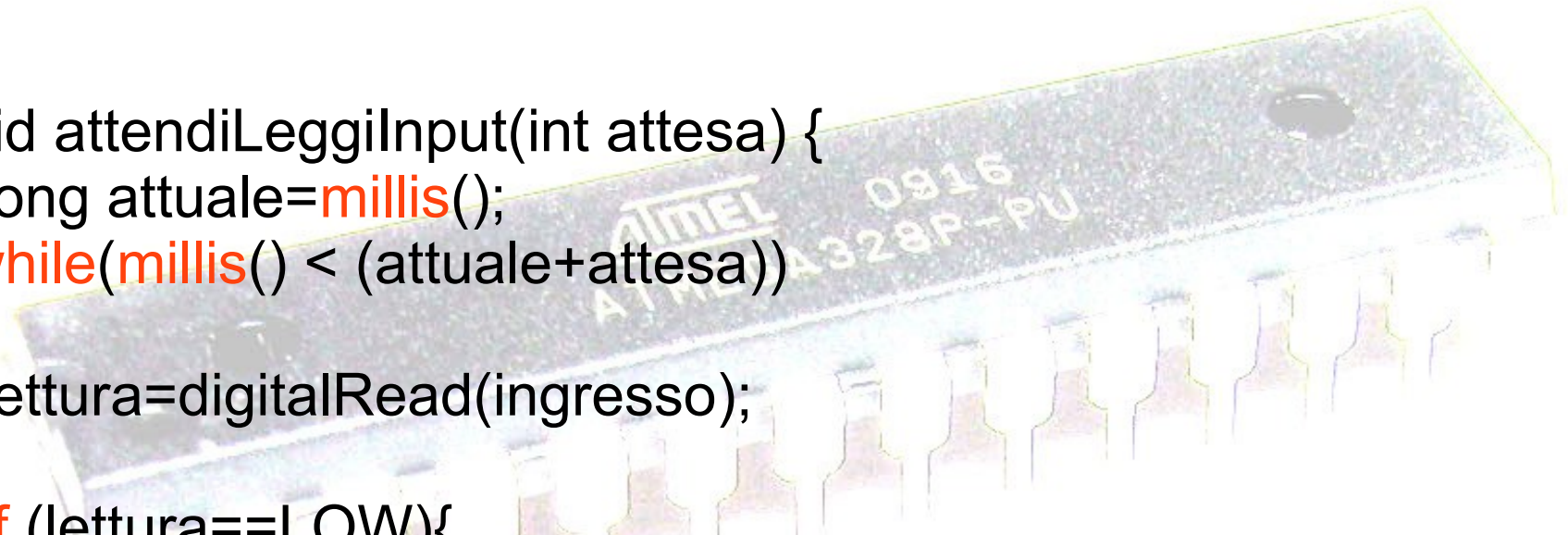
/* in questo modo la modifica al funzionamento può essere effettuata solo alla fine di una sequenza di lampeggio */

```
void loop() {
  if(lampeggio==false)
   normale();
   else
    gialloL();
   if (digitalRead(ingresso)==0)
       lampeggio=!lampeggio;
}
```



/* in questo modo la modifica al funzionamento può essere effettuata solo alla fine di una sequenza di lampeggio */

Sostituzione della delay per avere una risposta immediata



```
void attendiLeggiInput(int attesa) {
   long attuale=millis();
  while(millis() < (attuale+attesa))
  {
   lettura=digitalRead(ingresso);

   if (lettura==LOW){
      while(! digitalRead(ingresso)); // attende finchè resta basso
      delay(200);     // rimbalzo al rilascio
      lampeggio= ! lampeggio;
      break;   // esce dalla while
   }
  }
}
```

Utilizzo di attendiLeggiInput

```
void glalloL() {
  digitalWrite(giallo,HIGH);
  digitalWrite(verde,LOW);
  digitalWrite(rosso,LOW);
  //delay(1000);
  attendiLeggiInput(1000);
  if (! lampeggio) return;
  digitalWrite(giallo,LOW);
  digitalWrite(verde,LOW);
  digitalWrite(rosso,LOW);
  //delay(1000);
  attendiLeggiInput(1000);
}
```

sostituita

termina la funzione giallo

sostituita

Leggere un ingresso analogico e comunicarlo in seriale

Problema

Si vuole rilevare un input analogico, input da 0 a 5 Volt. Il convertitore ADC del microcontrollore lo converte in un numero intero da 0 a 1023. Tale numero verrà inviato in seriale al Pc.

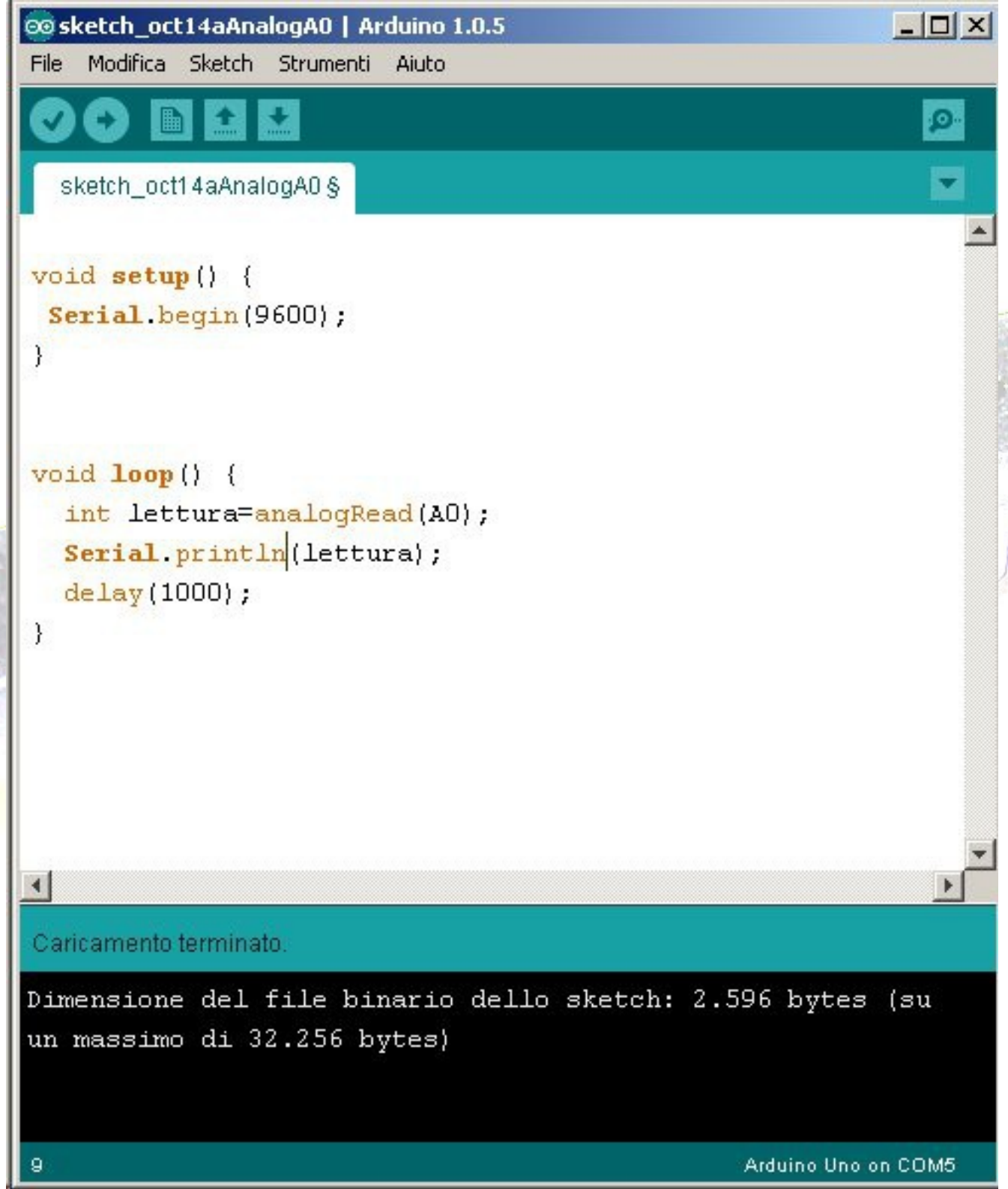
Soluzione

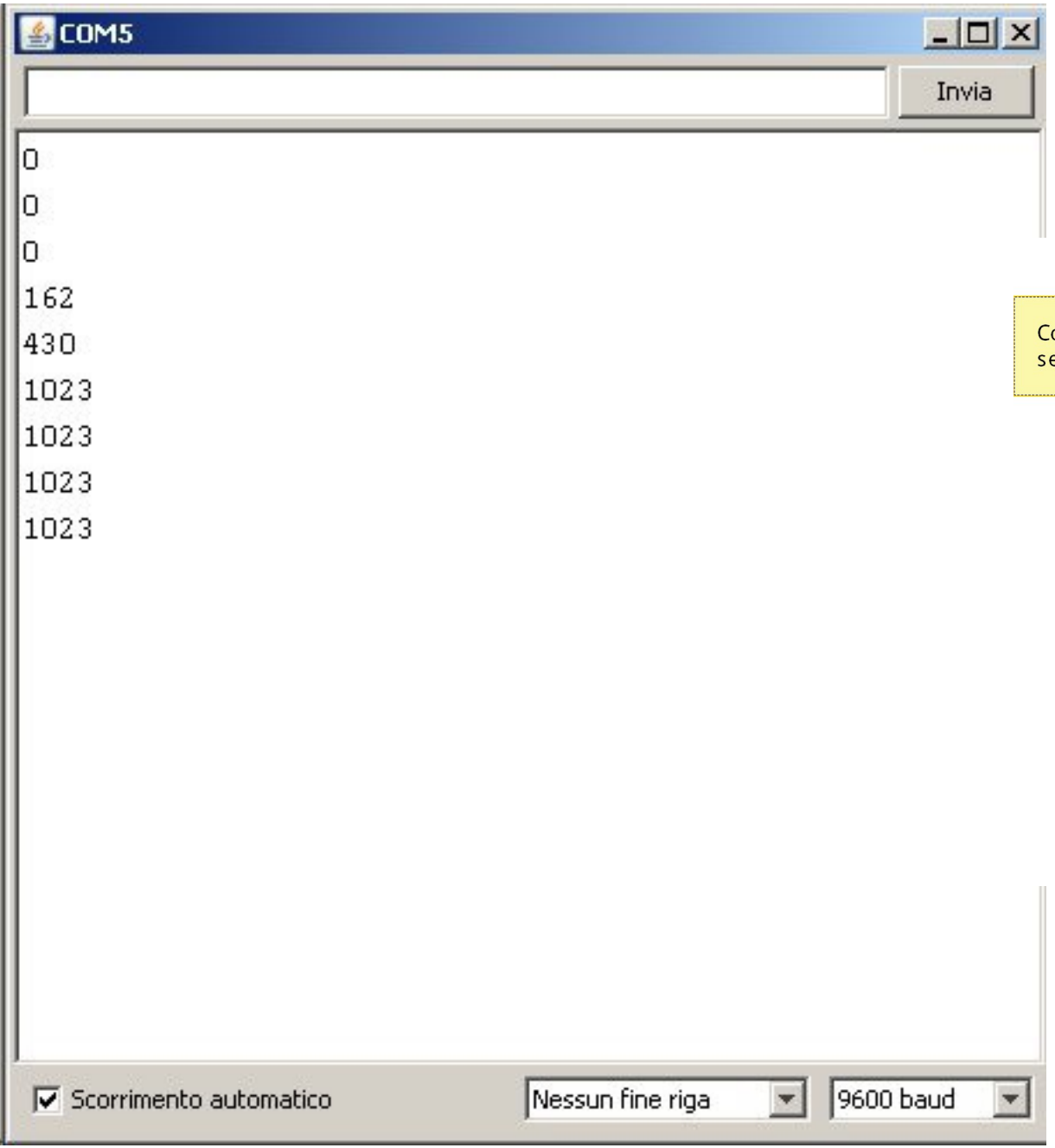
La lettura da input analogici viene effettuata con analogRead(A0-A5)

Per la comunicazione seriale si utilizza Serial.begin(9600) per inizializzare la comunicazione a 9600 bps e Serial.print(valore) per inviare un valore numerico sotto forma di caratteri codificati in ASCII al Pc.

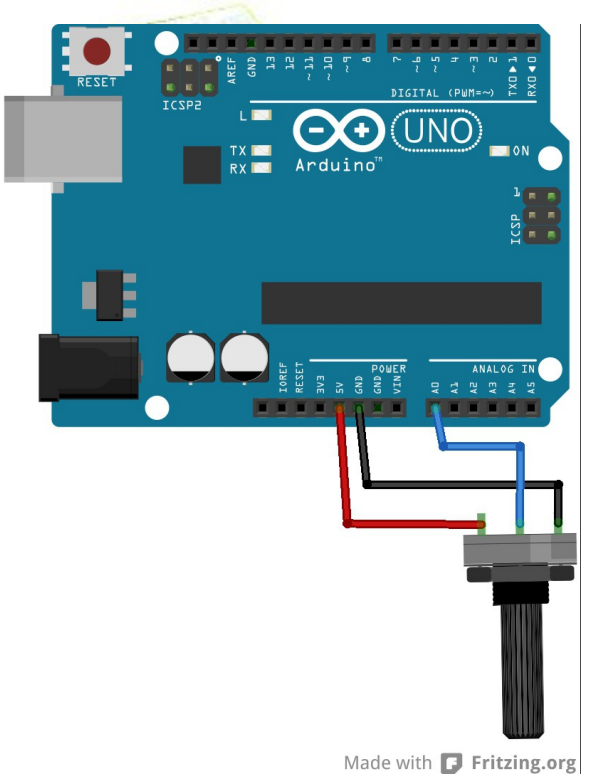
Se il dato fosse l'intero 123 i byte inviati sarebbero 49 50 51 corrispondenti ai codici ASCII rispettivamente di 1,2,3

La lettura sul Pc può essere eseguita sfruttando il Monitor seriale dell'IDE o con un programma specifico.


sketch_oct14aAnalogA0 | Arduino 1.0.5
File Modifica Sketch Strumenti Aiuto
sketch_oct14aAnalogA0 §
void setup() {
Serial.begin(9600);
}
void loop() {
int lettura=analogRead(A0);
Serial.println(lettura);
delay(1000);
}
Caricamento terminato.
Dimensione del file binario dello sketch: 2.596 bytes (su un massimo di 32.256 bytes)
9
Arduino Uno on COM5



Comunicazione seriale attraverso USB



Leggere più di 6 input analogici

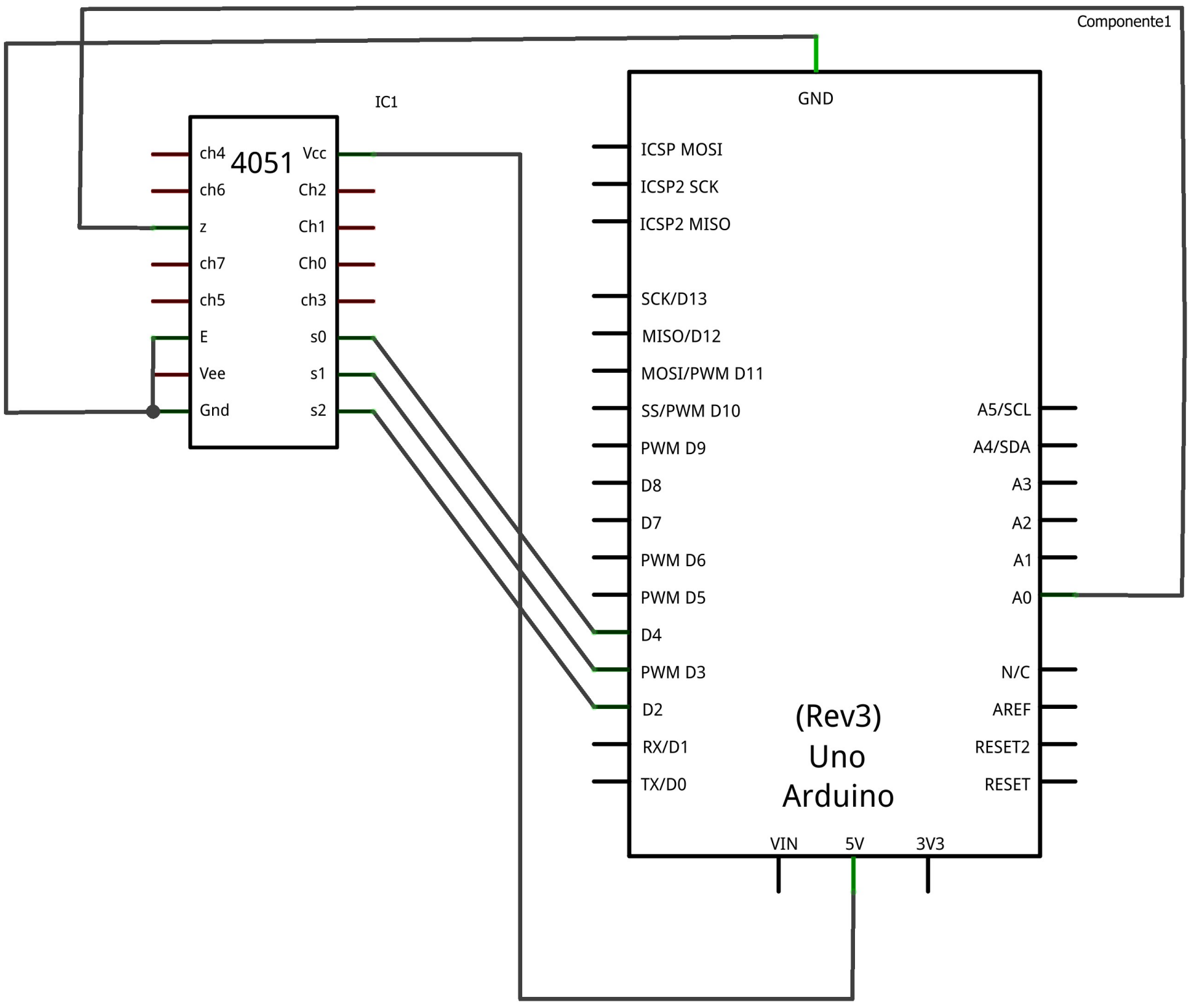
Problema

Si devono rilevare più input analogici di quanti pin analogici si abbiano a disposizione.

Le schede Arduino standard sono dotate di sei input analogici (le Mega di 16) e in alcuni casi gli input analogici non sono sufficienti per la propria applicazione. Per esempio, nella propria applicazione si può aver bisogno di regolare otto parametri girando le manopole di otto potenziometri.

Soluzione

Per selezionare e collegare diverse sorgenti di voltaggi su un solo input analogico, si deve utilizzare un chip multiplexer. Selezionando diverse sorgenti in modo sequenziale, si riesce a leggere una alla volta. Nell'esempio si utilizza il diffuso chip 4051, che si collega ad Arduino come mostrato in figura.



```
/* legge 8 valori analogici attraverso un unico pin di input analogico */

const int select[ ] = { 2,3,4};

const int analogPin =0;  // pin analogico collegato all'output del multiplexer

// la funzione restituisce il valore analogico di un canale

int getValue(int channel)

{

  // imposta i bit del selettore in modo che corrispondano al valore binario del canale

  for (int bit=0;bit <3; bit++)

  {  int pin=select[bit];

     int isBitSet = bitRead(channel,bit);

     digitalWrite(pin, isBitSet); }

  return analogRead(analogPin);

}
```

```
void setup()

{

  for (int bit=0; bit < 3;bit++)

     pinMode(select[bit], OUTPUT);

   Serial.begin(9600);

}

void loop()  {

// stampa i valori di ciascun canale una volta al secondo

  for (int channel=0; channel <8; channel++)

  {  int value = getValue(channel);

     Serial.print("Canale ");

     Serial.print(channel);

     Serial.print(" = ");

     Serial.print(value);   }

 Delay(1000);

}
```

/

Realizzare uno sketch con processing in modo che riceva dati seriali da Arduino

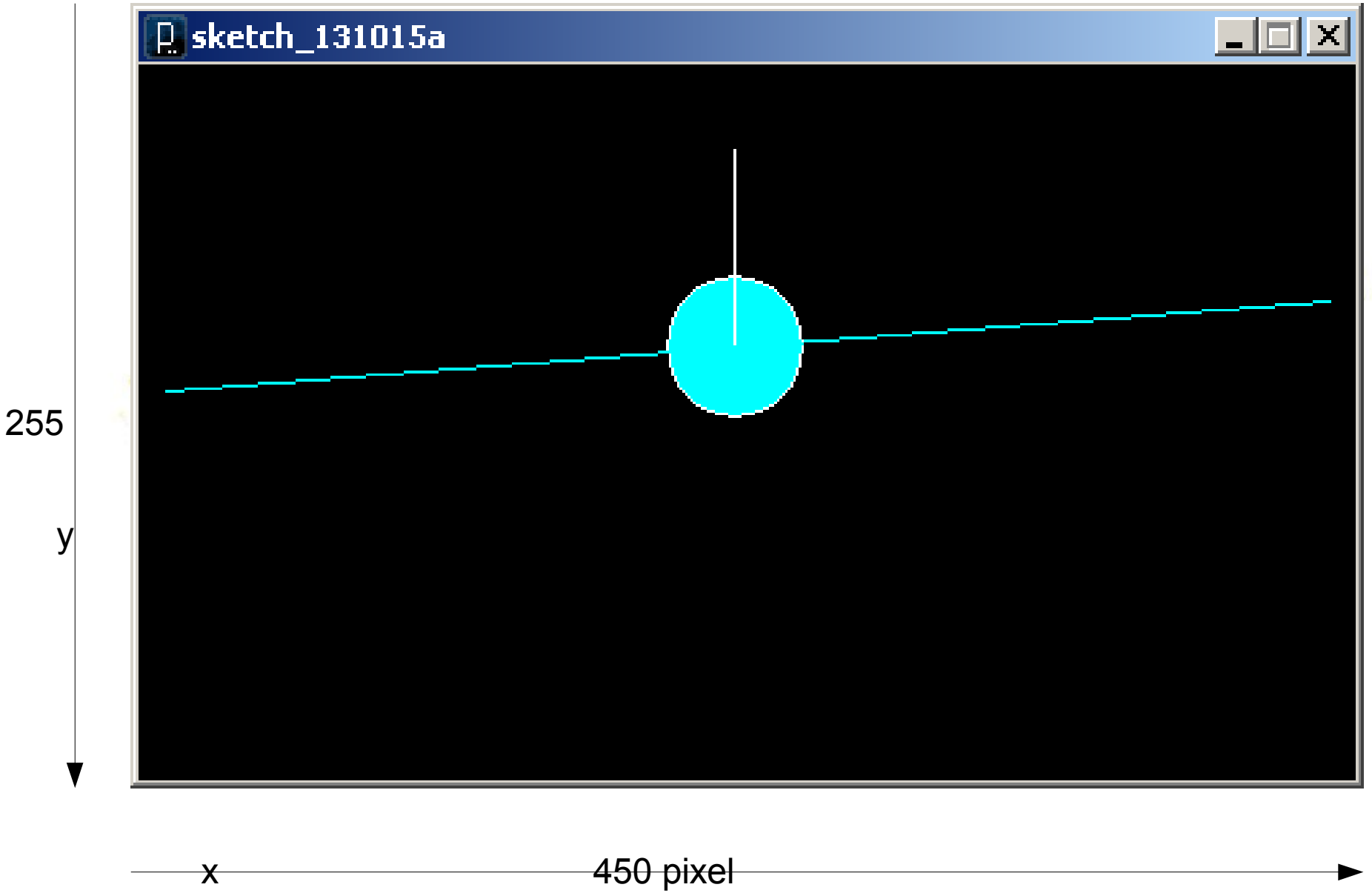
Problema

Si vuole utilizzare l'ambiente di sviluppo di Processing per ricevere in seriale da Arduino un valore corrispondente alla posizione di un potenziometro.
La posizione viene utilizzata per simulare l'inclinazione di un aereo visto di coda.

Soluzione

Lo sketch su Arduino invia ogni 300 ms un valore da 0 a 250 contenuto in un byte in base alla rotazione di un potenziometro collegato all'ingresso analogico A0.


sketch_oct12aSerialeAereo | Arduino 1.0.5
File Modifica Sketch Strumenti Aiuto
sketch_oct12aSerialeAereo
int posiz;
byte invio;
void setup()
{
Serial.begin(9600);
}
void loop() {
posiz=analogRead(A0);
invio=map(posiz,0,1023,0,250);
Serial.print(char(invio));
delay(300);
}
Caricamento terminato.
Dimensione del file binario dello sketch: 2.414 bytes (su un massimo di 32.256 bytes)
1
Arduino Uno on COM5


sketch_131015a
255
y
x
450 pixel

```
/**
 Il programma viene associato ad uno
sketch su arduino  che invia la posizione  di
un potenziometro, tramite lettura analogica.
 */

import processing.serial.*;

Serial serial2;
int data[]= new int[4];
int x1=10;
int y1=100;
int x2=440;
int y2=100;
int inByte;

void setup()
{

  size(450, 255);
  background(0); // nero
  stroke(0);
  serial2 = new Serial(this, Serial.list()[2],
9600);
  print(Serial.list()[2]);
}

void draw()
{ stroke(255);

  background(0);
  delay(200); // ritardo necessario
  fill(0,255,255);
  ellipse(220,100,50,50);/* cerchio
r=50 */
  line(220,100,220,30); //coda x=220
  if (serial2.available() > 0)
      inByte = serial2.read();
    data[0]=x1;
    data[2]=x2;
    data[1]=y1+(inByte-125);
    data[3]=y2-(inByte-125);
    /* Disegna la linea con le
coordinate ricevute. Se inByte= 125
rimane orizzontale */
    stroke(0,255,255);
    line(data[0], data[1], data[2],
data[3]); //ali

  }
```

Carica di un condensatore.

Problema

Un condensatore di capacità C, collegato alla tensione V_0 mediante una resistenza R, si carica secondo la legge esponenziale:

$$V = V_0\left(1 - e^{\frac{-t}{RC}}\right)$$

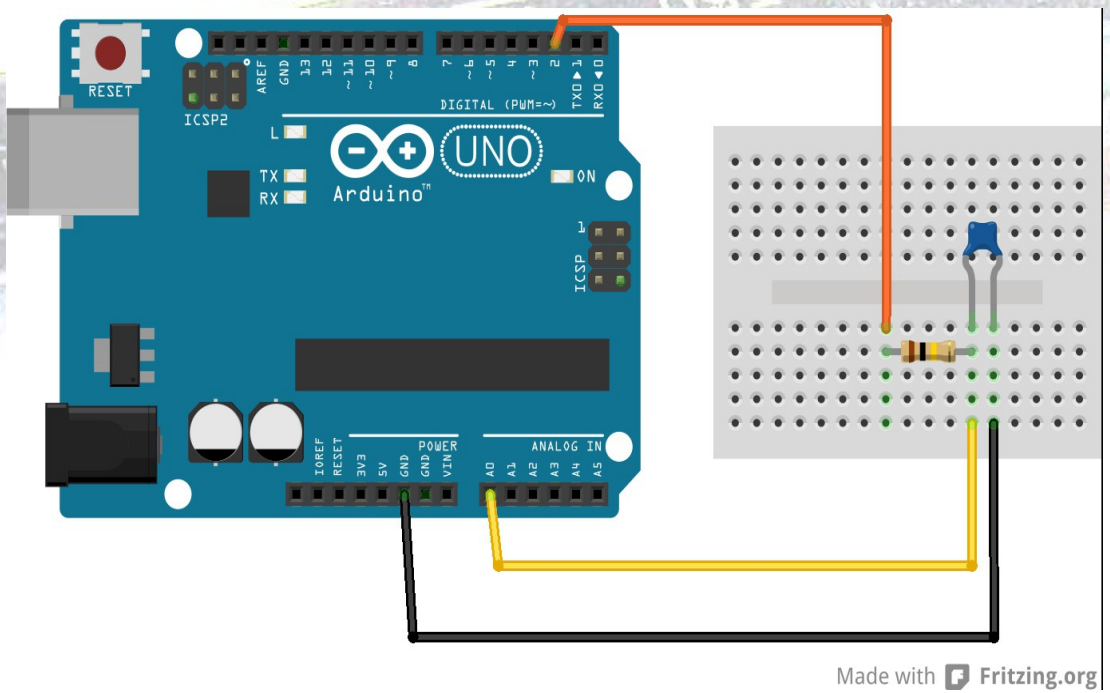
Il prodotto RC viene detto costante di tempo e dà un stima della velocità di carica del condensatore.

Dopo un tempo t=RC il condensatore si è caricato di circa il 63% (1- 1/*e*)del valore massimo. Si vuole verificare sperimentalmente la legge.

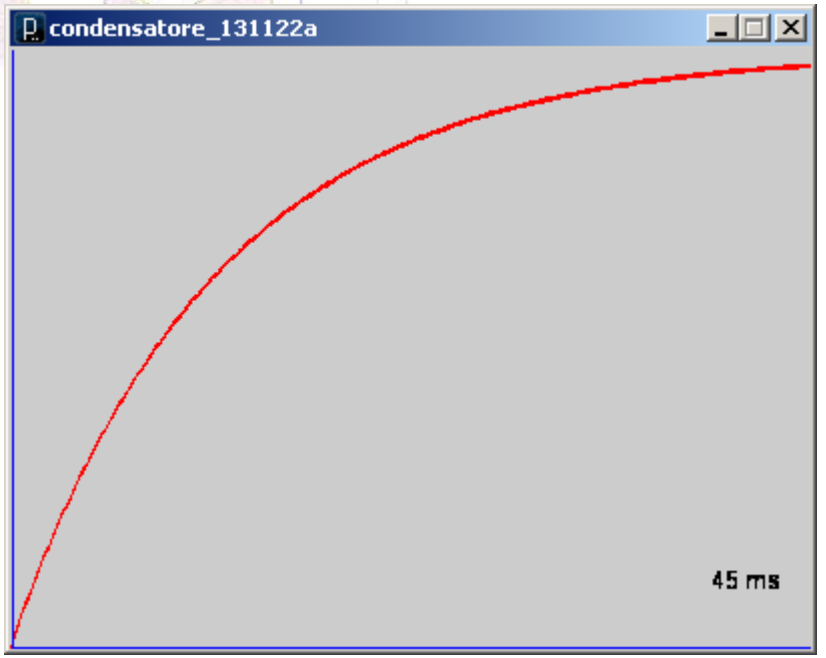
Soluzione

Lo sketch su arduino, dopo aver dato tensione al circuito, una resistenza e un condensatore collegati in serie, misura la tensione ai capi del condensatore in istanti successivi per un certo numero di volte, comunica i dati di tensione e di tempo ad uno sketch di processing che li visualizza su di un grafico.

R = 100 Kohm
C = 100 nF
RC= 0,01 s



```
#define NumMax 420
const int pinCarica =2;
const int led=13;
long inizio;
unsigned int tempo[NumMax];
int lettura[NumMax];

void setup() {
   Serial.begin(9600);
   pinMode(pinCarica,OUTPUT);
   pinMode(led,OUTPUT);
}


void loop() {
  digitalWrite(pinCarica, LOW); // scarica
condensatore
  delay(1000);
  digitalWrite(led,HIGH); // segnalazione
  digitalWrite(pinCarica,HIGH); // inizio carica
  inizio=micros();
  for (int k=0;k<NumMax;k++) {
    tempo[k]=micros()- inizio;
    lettura[k]=analogRead(A0); // 113 us
  }
// invio dati in seriale
 for(int k=0;k<NumMax;k++) {
   Serial.print(lettura[k]);
   Serial.print(' ');
   Serial.print(tempo[k]/10);
   Serial.print(' ');
 }
 Serial.print('*');
 digitalWrite(led,LOW); // fine
segnalazione
 delay(10000);
}
```



R=120 Kohm C=100 nF

```
import processing.serial.*;

String myString = null;
boolean fine=false;
// The serial port:
Serial myPort;



void setup() {
  size(400,300);
  myPort = new Serial(this, Serial.list()
[2], 9600);
}



void serialEvent(Serial myPort) {
  char carattere=myPort.readChar();
  myString=myString+carattere;
  if(carattere=='*') fine=true;
}
```

```
void draw() {
  stroke(0,0,255);
  line(1,299,399,299);
  line(1,299,1,1);
  fill(0);     // nero
  text("45 ms", 350,270);
   if (fine)
   {
    print(myString);
    fine=false;
    int[] dati= int(split(myString,char(0x20)));
    stroke(255,0,0);
    int x= 0;
    int y= 299;

    if(dati != null) {
    for(int k=0;k<dati.length-1;k=k+2){
      int x1=dati[k+1]/10;
      int y1=299-(dati[k]*299/1023);
      line(x,y,x1,y1);
      x=x1;
      y=y1;
    }
    myString="";
    }
    }
}
```

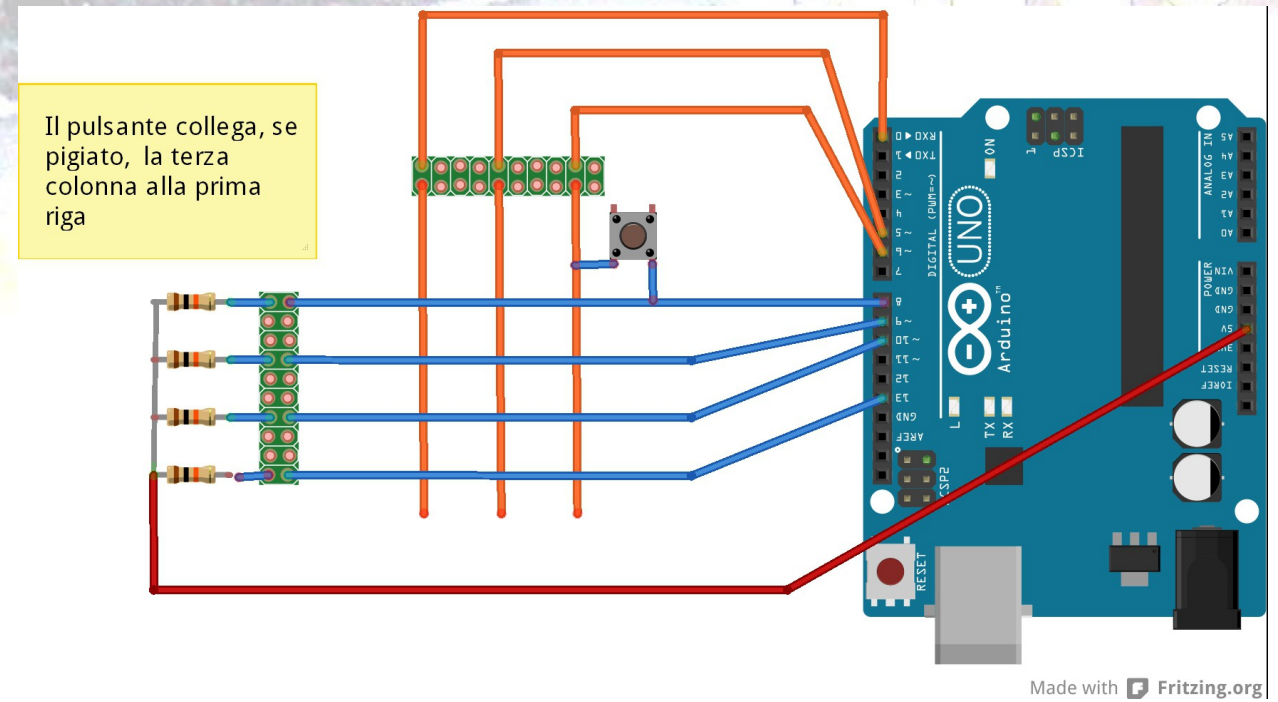
Leggere un tastierino a matrice

Problema

Si vuole rilevare la pressione dei tasti di un tastierino a matrice righe, colonne come quello dei telefoni.

Soluzione

Si devono collegare le 4 righe e le 3 colonne del connettore del tastierino rispettivamente a 4 ingressi e a 3 uscite digitali di Arduino come in figura. Quando viene pigiato un tasto, viene messa in comunicazione una colonna con una riga. In figura per semplicità è indicato un solo tasto.



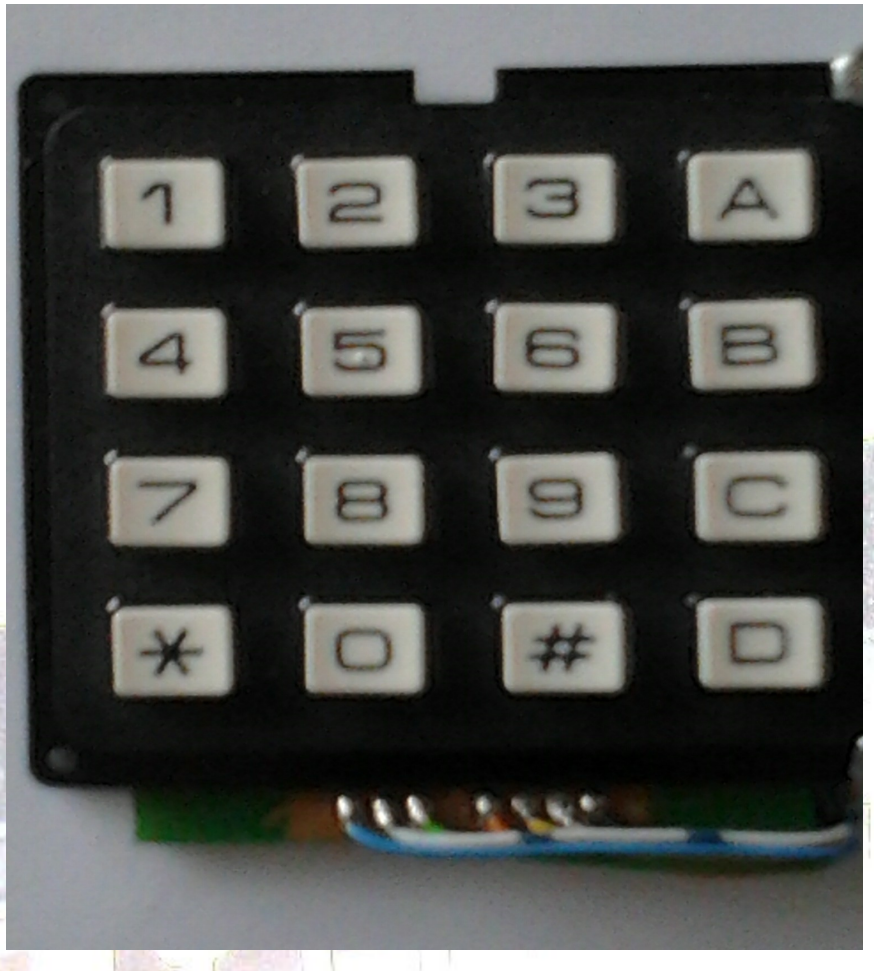
Discussione

Le tastiere a matrice sono di solito costituite da pulsanti normalmente aperti che, quando vengono premuti, collegano una riga a un colonna. La funzione **getKey i**mposta in sequenza su **LOW** i pin per ciscuna colonna e poi controlla se i pin di alcune righe sono **LOW.** Dato che si utilizzano delle resistenze di pull-up le righe saranno **HIGH** a meno che non si chiuda un pulsante.

```
const char keymap[4][3]= {
  {'1', '2', '3'} ,
  {'4', '5', '6'},
  {'7', '8', '9'},
  {'*', '0', '#'}
};
int numCar=0;
int offset=0;
int tempTime=0;
int time;
const int rowPins[4]={8,9,10,13};
const int colPins[3]= {0,6,7};
```

```
char getKey() {
  char key=0;
  for(int column=0;column<3;column++)
  {
    digitalWrite(colPins[column],LOW);
    for(int row=0;row<4; row++)
    {
      if(digitalRead(rowPins[row])==LOW)
      {
        delay(20);
        while(digitalRead(rowPins[row])== LOW);
        key= keymap[row][column];
      }
    }
   digitalWrite(colPins[column],HIGH);
  }
  return key;
}
```

```
void setup()
{ Serial.begin(9600);

  for (int row=0;row<4;row++)
  {
    pinMode(rowPins[row],INPUT);
    digitalWrite(rowPins[row],HIGH); //attiva pull-up
  }
  for (int column=0;column<3;column++)
    {
      pinMode(colPins[column],OUTPUT);
      digitalWrite(colPins[column],HIGH); //colonne tutte inattive
    }
}

void loop() {
   char key=getKey();
   if(key != 0) {
     Serial.println(key);
}
```



Contatti da sx: colonne (la prima è 1,4,7,*) righe (la prima è 1,2,3,A)
La quarta colonna non è utilizzata.

Utilizzare display

Collegare e utilizzare un display LCD alfanumerico

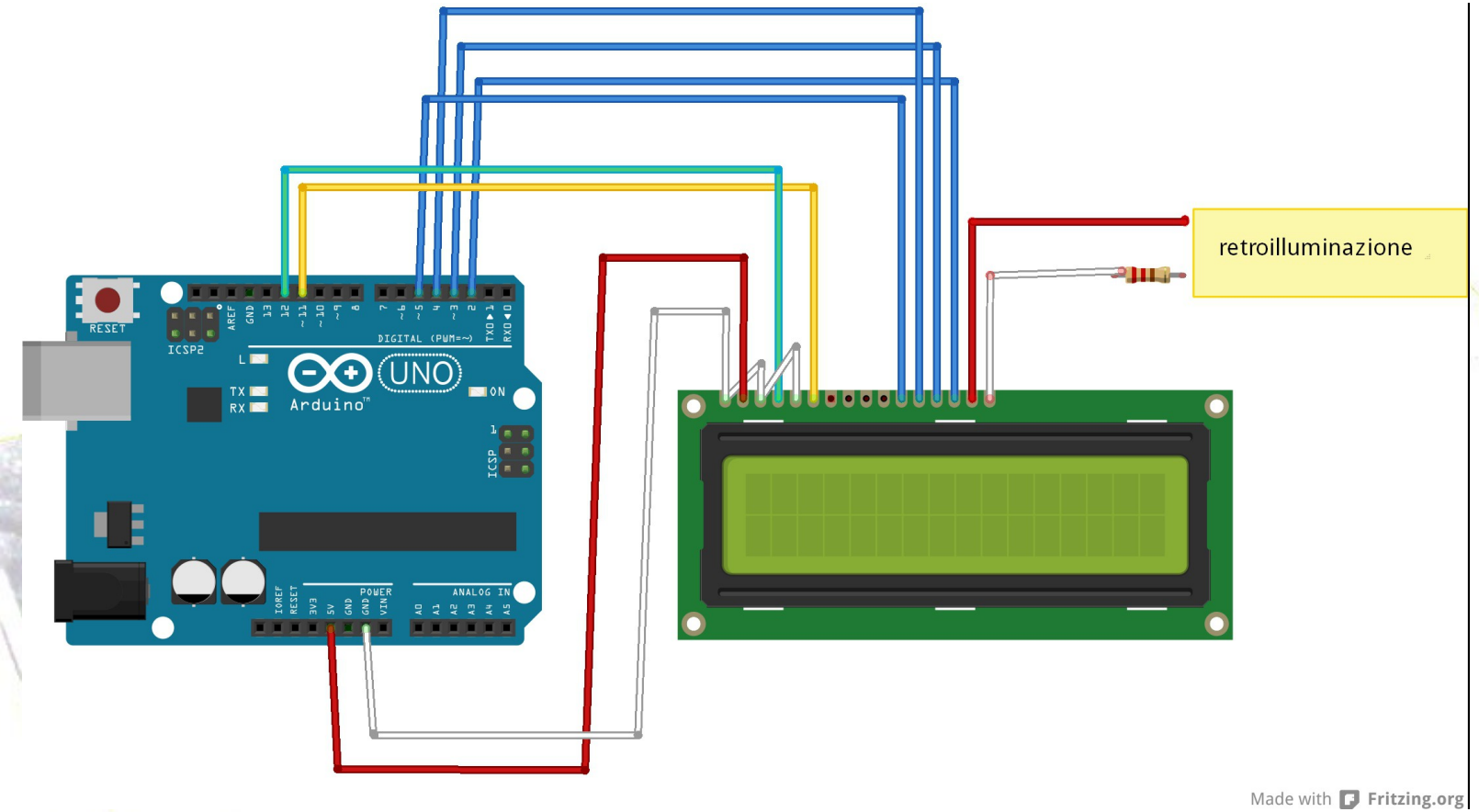
Problema

Si vuole utilizzare un LCD alfanumerico basato sul controller HD44780 di Hitachi o un chip compatibile e si vogliono visualizzare valori alfanumerici.

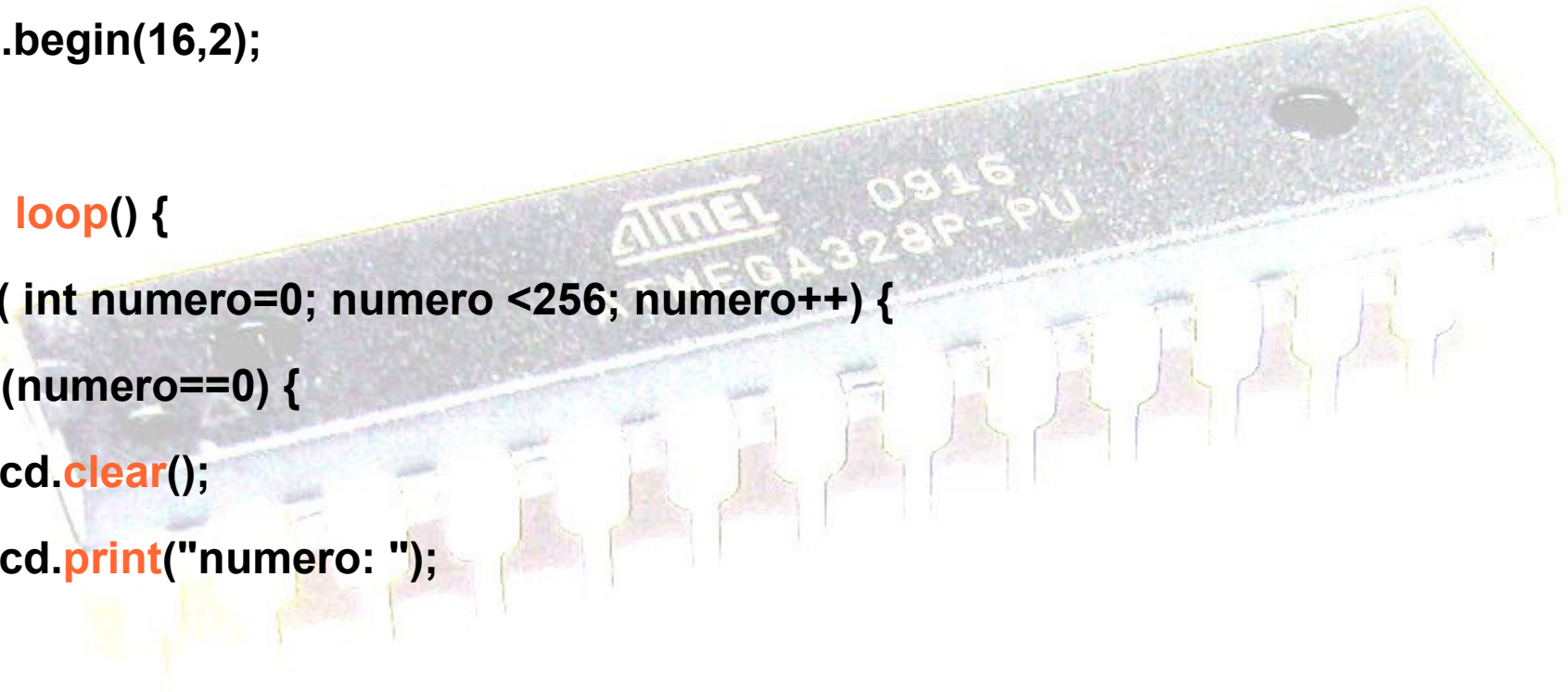
Soluzione

Il software di Arduino include la libreria LiquidCrystal, che serve a gestire i dispositivi LCD basati sul chip HD44780.

Pin del display LCD	Funzione	Pin di Arduino
1	GND o Vss	GND
2	+5V o Vdd	+5V
3	contrasto	Da 0 a +5V
4	RS	12
5	R/W	GND
6	E	11
7	D0	
8	D1	
9	D2	
10	D3	
11	D4	5
12	D5	4
13	D6	3
14	D7	2
15	+5 retroilluminazione	
16	GND retroilluminazione	


retroilluminazione
RESET
ICSP2
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM=~)
L
TX
RX
UNO
Arduino™
ON
ICSP
POWER
ANALOG IN
IOREF
RESET
3V3
5V
GND
GND
VIN
A0
A1
A2
A3
A4
A5
Made with Fritzing.org

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(12,11,5,4,3,2);    // RS E D4 D5 D6 D7
void setup()
{
  lcd.begin(16,2);
 }
void loop() {
  for( int numero=0; numero <256; numero++) {
    if (numero==0) {
      lcd.clear();
      lcd.print("numero: ");
    }
    lcd.setCursor(9,0);
   lcd.print(numero);
    delay(200);
   }
  }
```



Collegare e utilizzare display a 7 segmenti

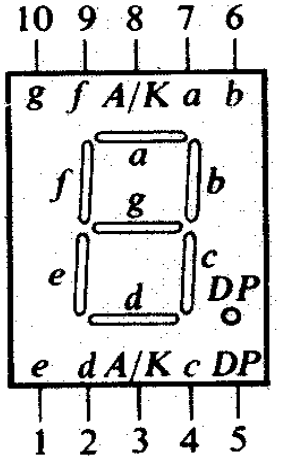
Problema

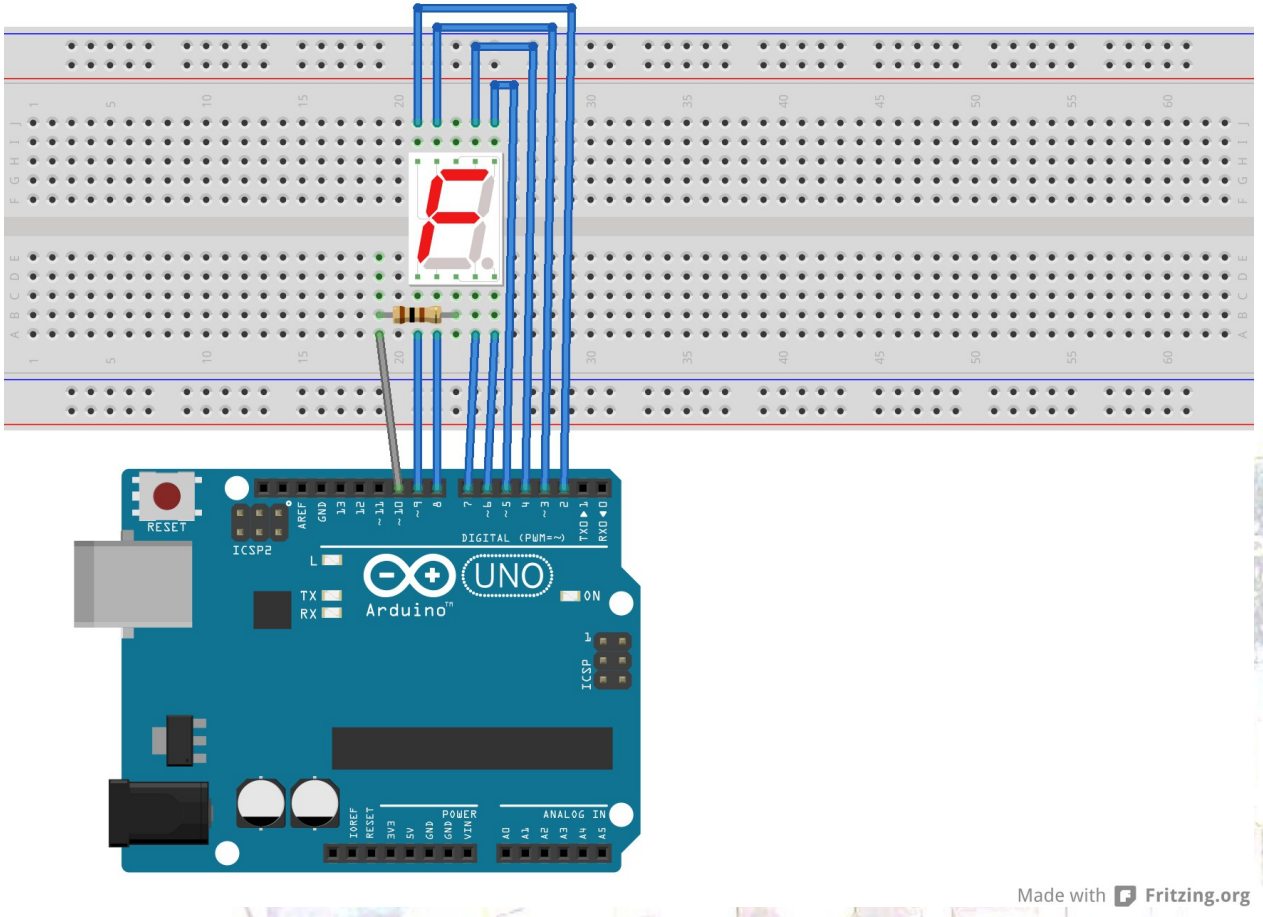
Si vogliono utilizzare dei display a 7 segmenti per visualizzare dei numeri.

Soluzione

Un display a 7 segmenti ha la piedinatura che si vede in figura. Se il display è a catodo comune questo andrà collegato a massa attraverso una resistenza.

Gli altri pin denominati a,b,c,d.e,
f,g possono essere collegati
come nello schema seguente.



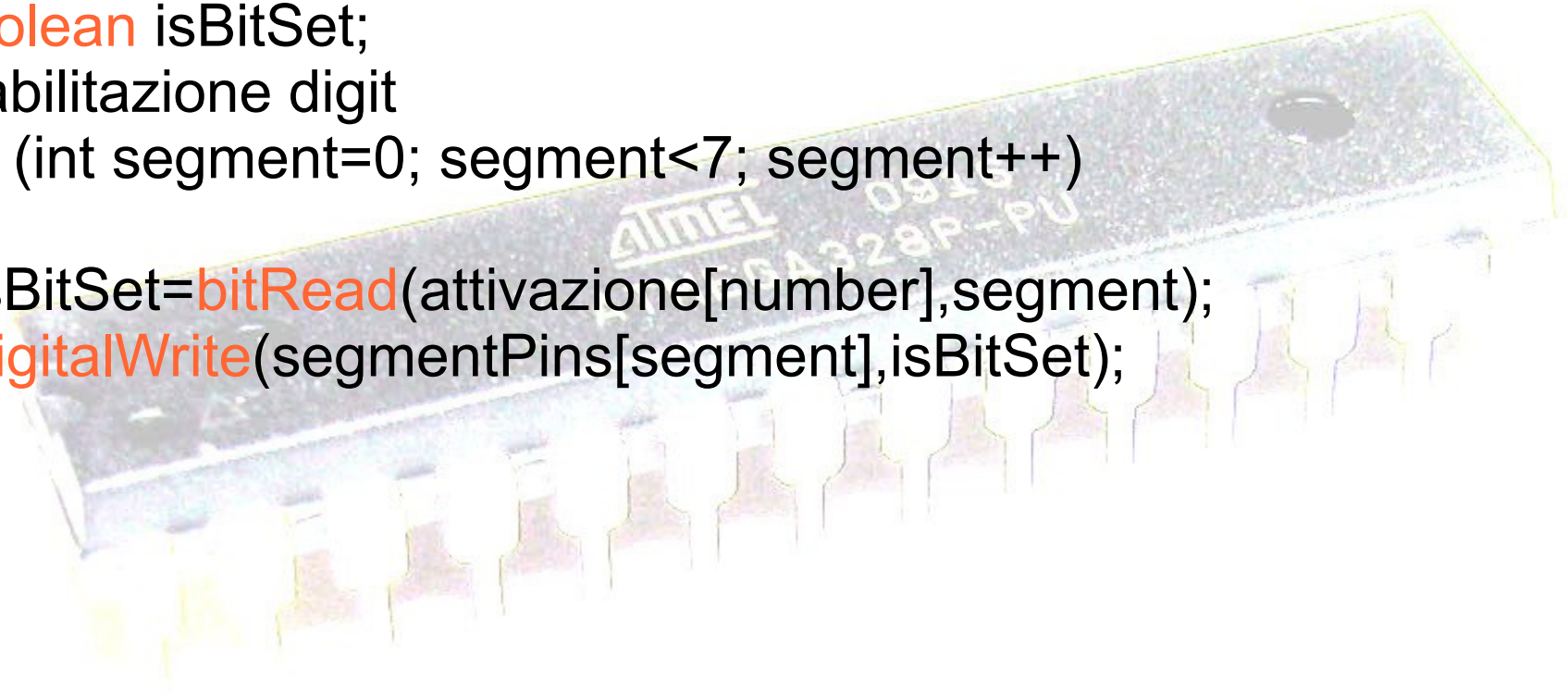


Display	a	b	c	d	e	f	g
Pin	4	5	7	8	9	3	2

Nell'esempio che segue viene utilizzato un unico display. Il catodo comune del display non è collegato a massa ma al pin 10 della scheda Arduino. Tale pin viene dichiarato in output e attivato al valore **LOW**.

```
const int segmentPins[7]={4,5,7,8,9,3,2};   // a b c d e f g
const int catodo =10;                       // segmentPin[0] ⟶ a
const int attivazione[10]={0b00111111,0b00000110,0b01011011,
0b01001111,0b01100100,0b01101101,0b01111101,0b00000111,0b01111
111,0b01101111};        // 0gfedcba

void setup() {
 for (int i=0;i<8;i++)
   pinMode(segmentPins[i],OUTPUT);

   pinMode(catodo,OUTPUT);
   digitalWrite(catodo,LOW);    // abilitazione del display
}
void loop() {                          // cifre da 0 a 9 in sequenza
  for (int numero=0; numero<10;numero++)
  {
  showDigit(numero);
  delay(1000);
  }
}
```

```
void showDigit(int number) {
  boolean isBitSet;
  // abilitazione digit
  for (int segment=0; segment<7; segment++)
  {
    isBitSet=bitRead(attivazione[number],segment);
    digitalWrite(segmentPins[segment],isBitSet);
  }
}
```



Se i display sono più di uno si può utilizzare la tecnica del multiplexing.

Tutti i sette pin a,b.c,d,e,f,g dei display sono collegati in parallelo mentre i catodi comuni sono attivati da altri pin di Arduino.

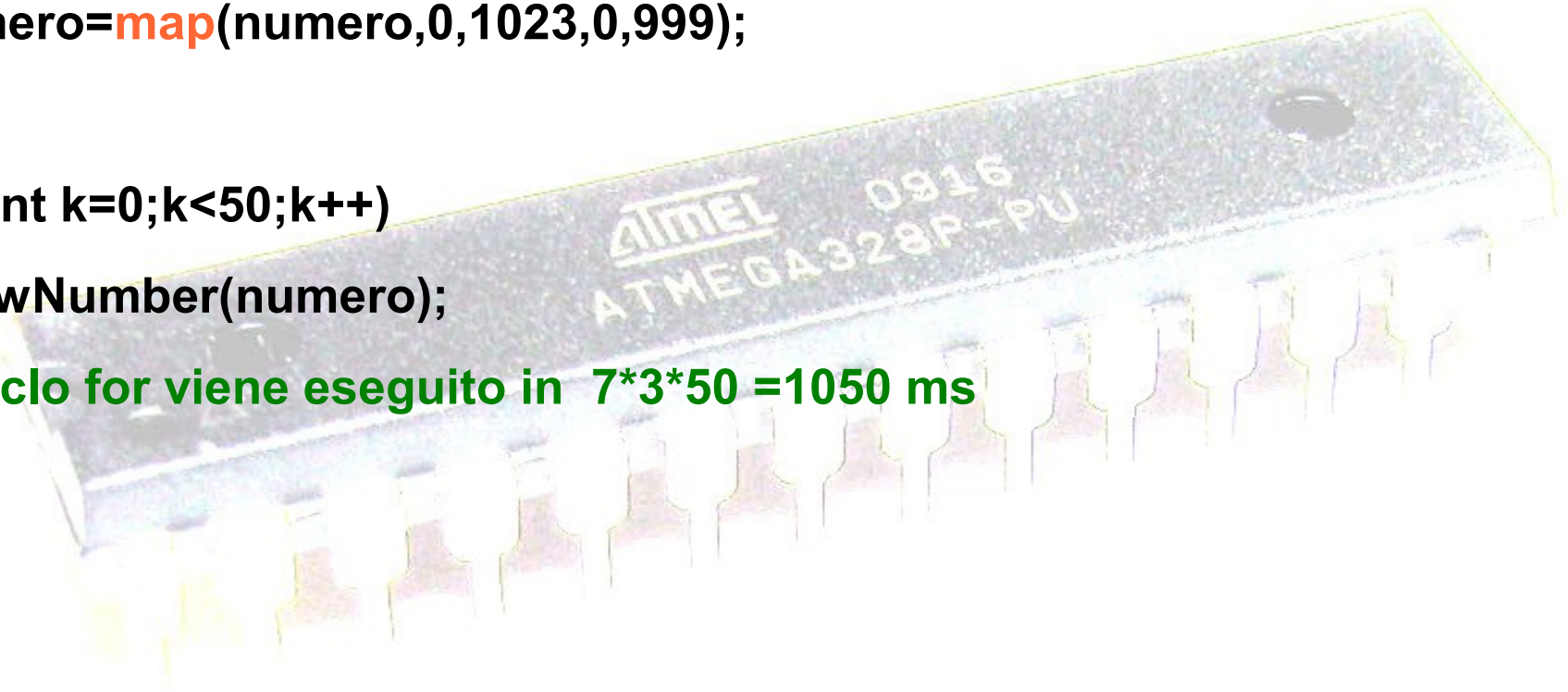
L'attivazione avviene con valore logico 0 e viene fatta ciclicamente per i vari display.

La cosa importante è che i display vengano attivati tutti in un tempo totale di circa 20 ms. In questo modo, per la persistenza della visione, l'impressione è che tutti i display siano accesi contemporaneamente.

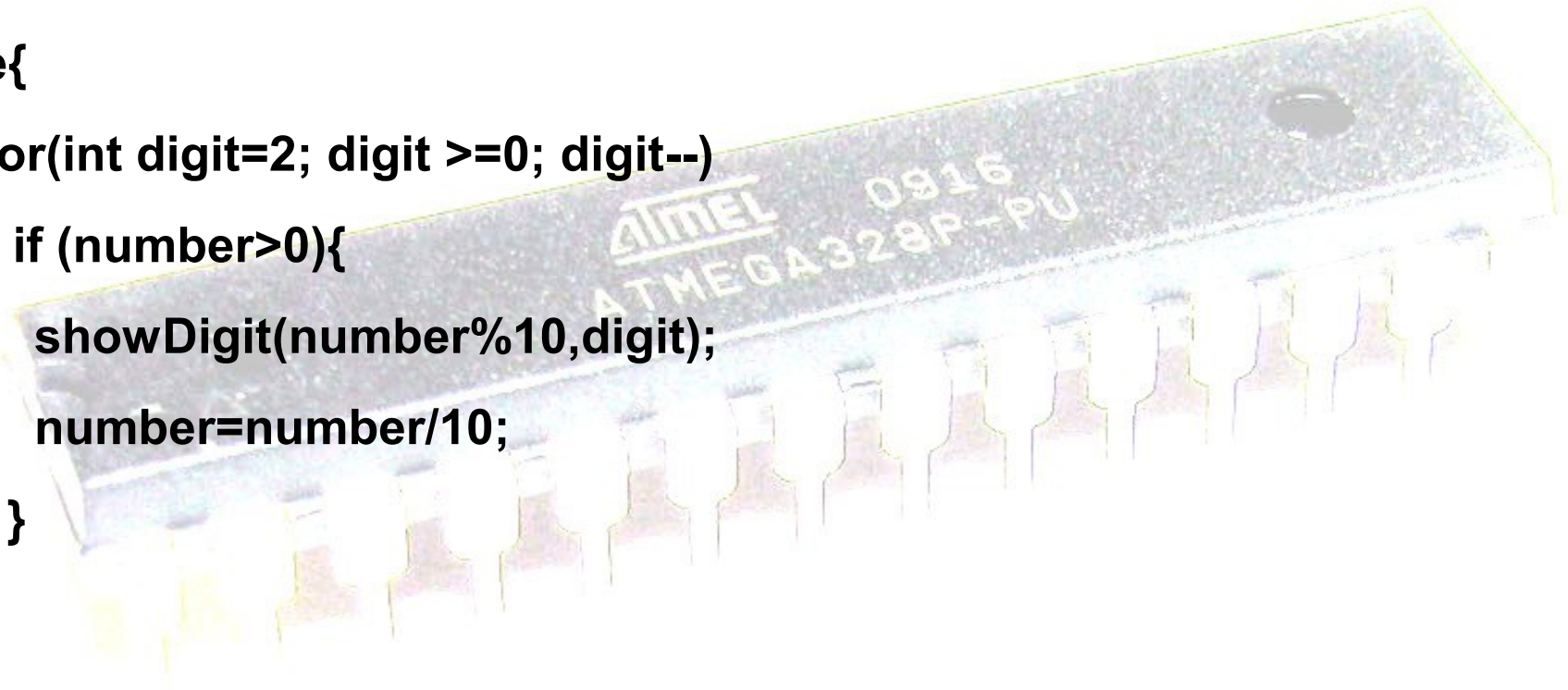
Quando si attiva un display viene inviata ai suoi led la configurazione binaria opportuna, per visualizzare la cifra numerica.

```
const int segmentPins[7]={4,5,7,8,9,3,2}; // a b c d e f g

const int digitPins[3]={10,11,12};

const int numerale[10]={0b00111111,0b00000110,0b01011011,

0b01001111,0b01100100,0b01101101,0b01111101,0b00000111,0b01111111,0b0
1101111};

void setup() {

 for (int i=0;i<8;i++)

   pinMode(segmentPins[i],OUTPUT);

 for (int i=10;i<13;i++)

   pinMode(i,OUTPUT);

    for(int i=0;i<3;i++)

      digitalWrite(digitPins[i],HIGH);

}
```

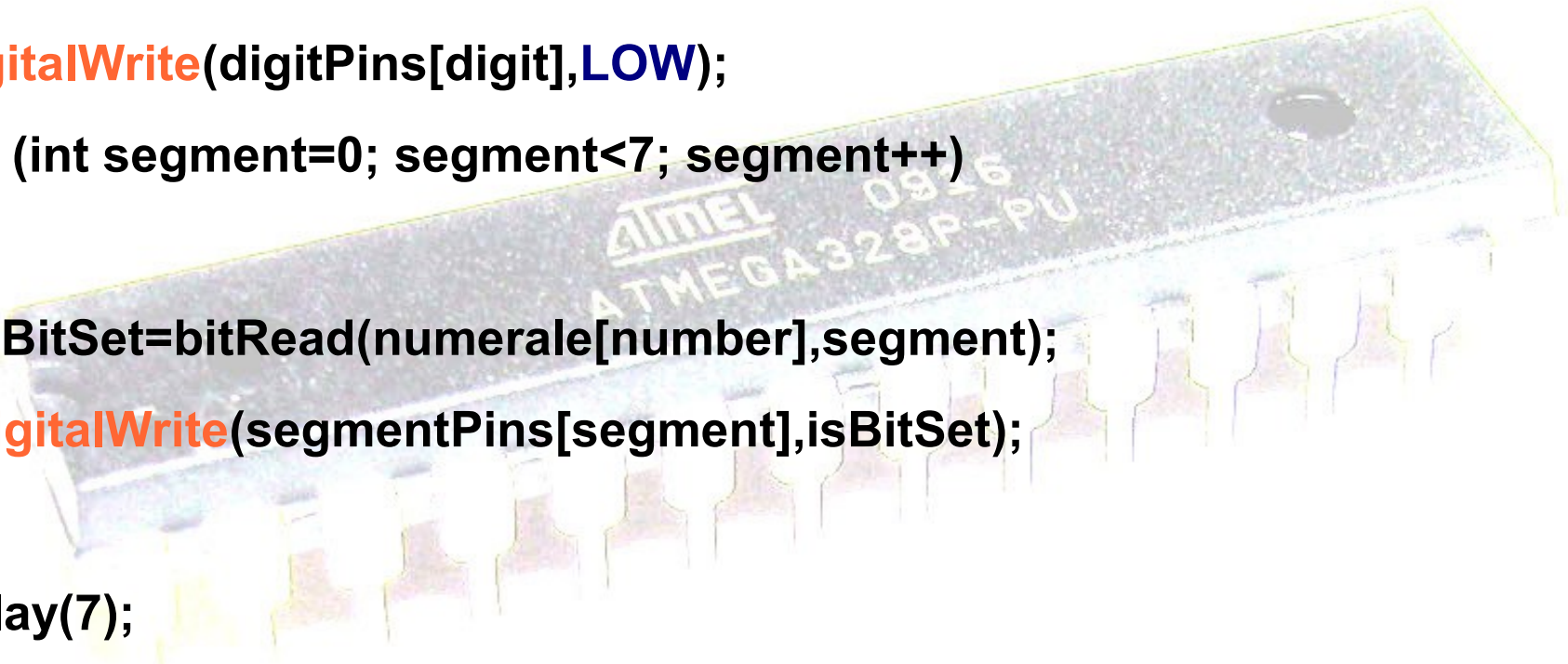
```
void loop() {
  int numero=analogRead(A0);
  numero=map(numero,0,1023,0,999);

  for(int k=0;k<50;k++)
  showNumber(numero);
// il ciclo for viene eseguito in  7*3*50 =1050 ms
}
```



```
// richiama 3 volte la showDigit
void showNumber(int number) {
  if (number==0)
    showDigit(0,2);  // showDigit( cifra, display)
else{
    for(int digit=2; digit >=0; digit--)
    { if (number>0){
       showDigit(number%10,digit);
       number=number/10;
      }
    }
  }
```



```
// ritarda 7 millisecondi
void showDigit(int number,int digit) {
  boolean isBitSet;
  // abilitazione digit
  digitalWrite(digitPins[digit],LOW);
  for (int segment=0; segment<7; segment++)
  {
    isBitSet=bitRead(numerale[number],segment);
    digitalWrite(segmentPins[segment],isBitSet);
  }
  delay(7);
  digitalWrite(digitPins[digit],HIGH);
}
```



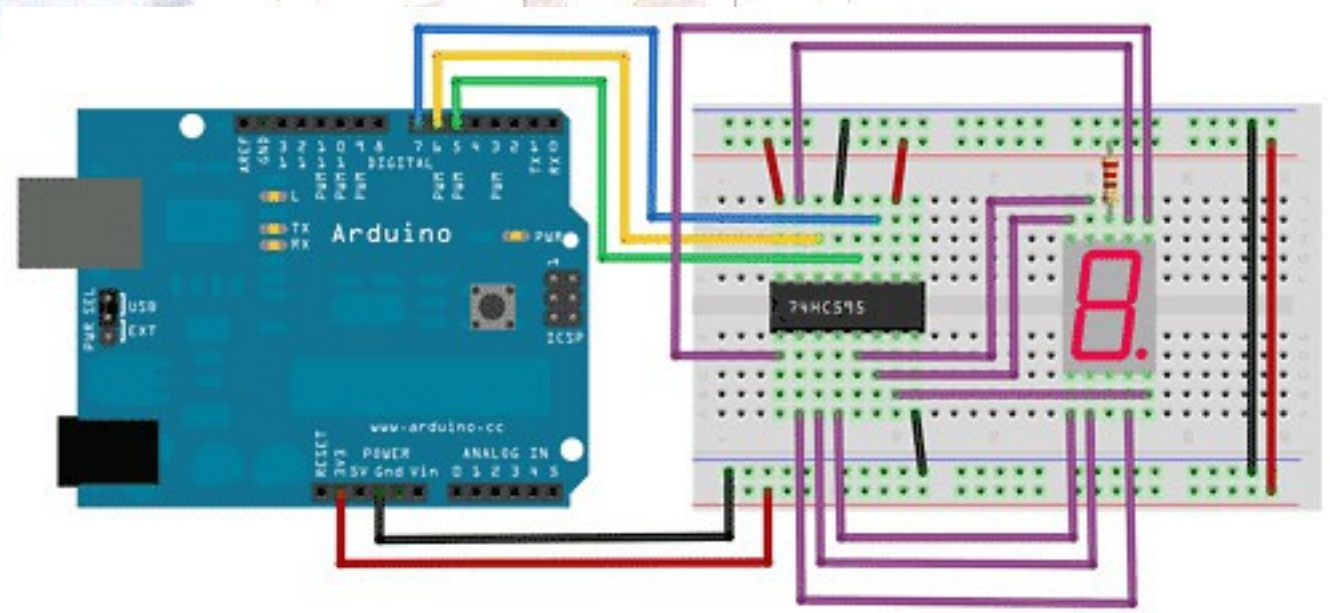
Utilizzare una comunicazione seriale per pilotare un display a 7 segmenti

Problema

Si vuole controllare un display a 7 segmenti inviando i dati con una comunicazione seriale sincrona .

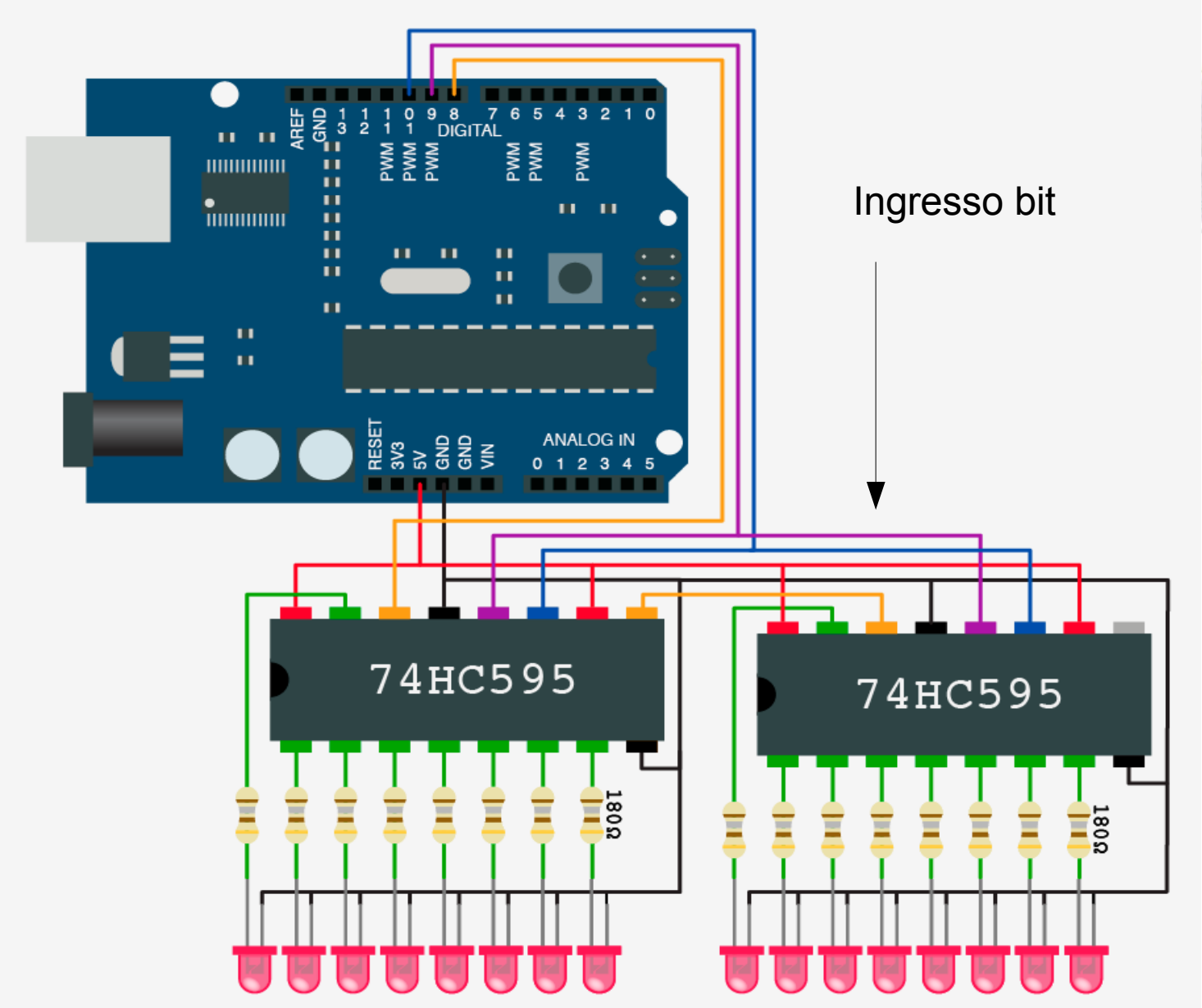
Soluzione

Si utilizza uno shift register come 74HC595 e la funzione shiftOut per inviare 8 bit in modo seriale sincrono (con clock).
In questo modo si utilizzano solo 3 pin digitali di Arduino, uno per i bit inviati in serie, uno per il clock e uno per l'abilitazione dell'uscita verso il display quando il dato (byte) è pronto



```
int latchPin = 5;  //Pin connected to ST_CP of 74HC595 invio in uscita
int clockPin = 7; //Pin connected to SH_CP of 74HC595 clock
int dataPin = 6;  //Pin connected to DS of 74HC595 dati

const int cifra[16]=   //0gfedcba
{0b00111111,0b00000110,0b01011011,0b01001111,0b01100100,0b01101101,0b01111101,0b0
0000111,
0b01111111,0b01101111,0b01110111,0b01111100,0b00111001,0b01011110,0b01111001,0b01
10001};

void setup() {
  pinMode(latchPin, OUTPUT); //set pins to output
  pinMode(clockPin, OUTPUT);
  pinMode(dataPin, OUTPUT);
}

void loop() {

 for(int k=0;k<16;k++) {
    //ground latchPin and hold low for as long as you are transmitting
    digitalWrite(latchPin, LOW);
    shiftOut(dataPin, clockPin, MSBFIRST,cifra[k]);   /* per lo zero si invia 00111111 dal più
significativo al termine dello shift sull'uscita q0 del 74HC595 ci sarà il bit meno
significativo, in questo caso 1*/
   digitalWrite(latchPin, HIGH);
   delay(1000);
  }
}
```

Collegamento in cascata di shift register 74HC595



Accensione dei led come supercar

```
void loop() {

  for (int j = 1; j < 65536; j=j*2) {

    digitalWrite(latchPin, LOW);
    shiftOut(dataPin, clockPin, MSBFIRST, (j>>8));  // invio 8 bit alti
    shiftOut(dataPin,clockPin,MSBFIRST,j);          // invio 8 bit bassi

    digitalWrite(latchPin, HIGH);

    delay(30);
  }

    for (int j = 32768; j >0 ; j=j/2) {

    digitalWrite(latchPin, LOW);
    shiftOut(dataPin, clockPin, MSBFIRST, (j>>8));
    shiftOut(dataPin, clockPin, MSBFIRST, j);

    digitalWrite(latchPin, HIGH);

    delay(30);
  }
}
```

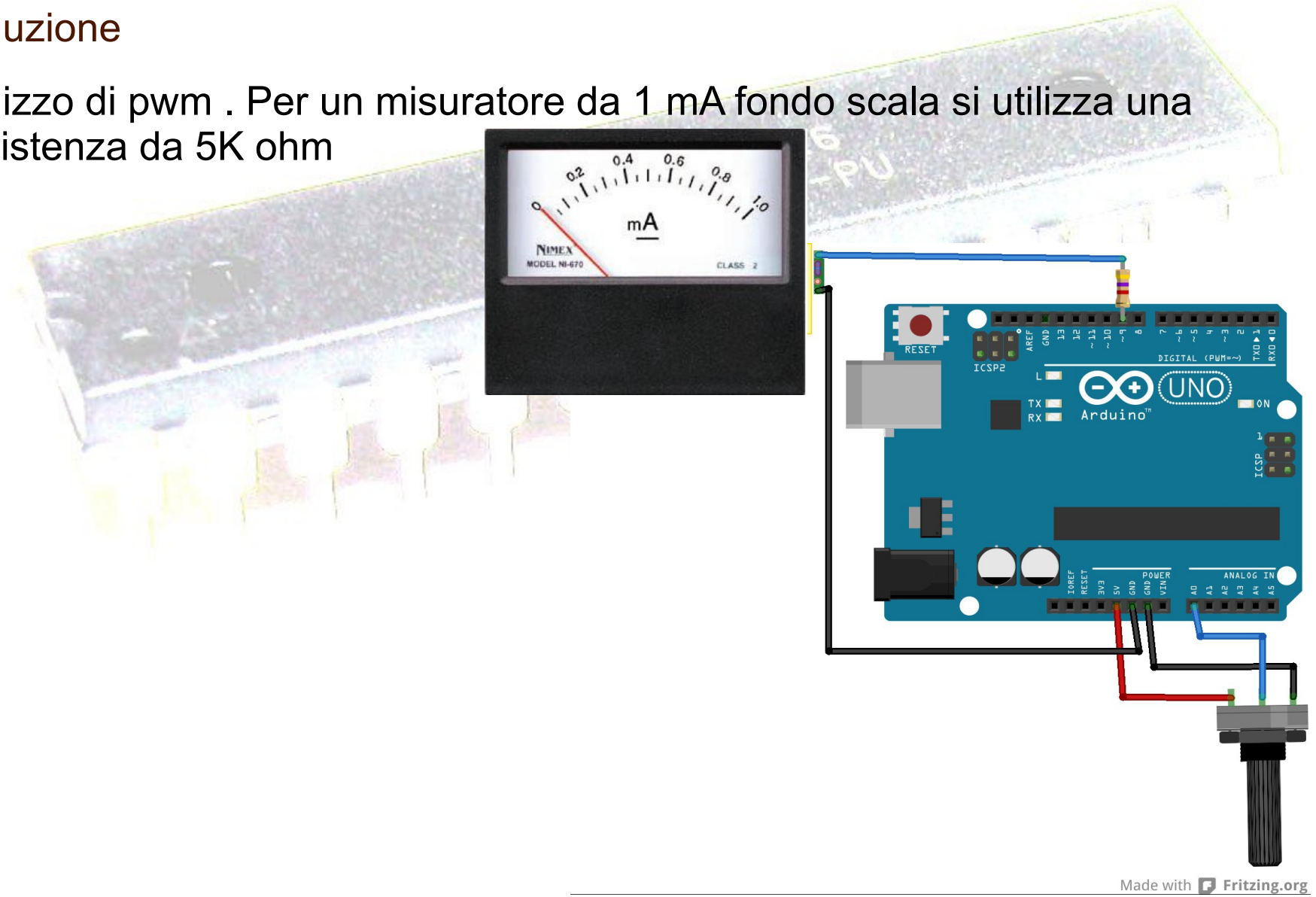
Utilizzare come display uno strumento analogico

Problema

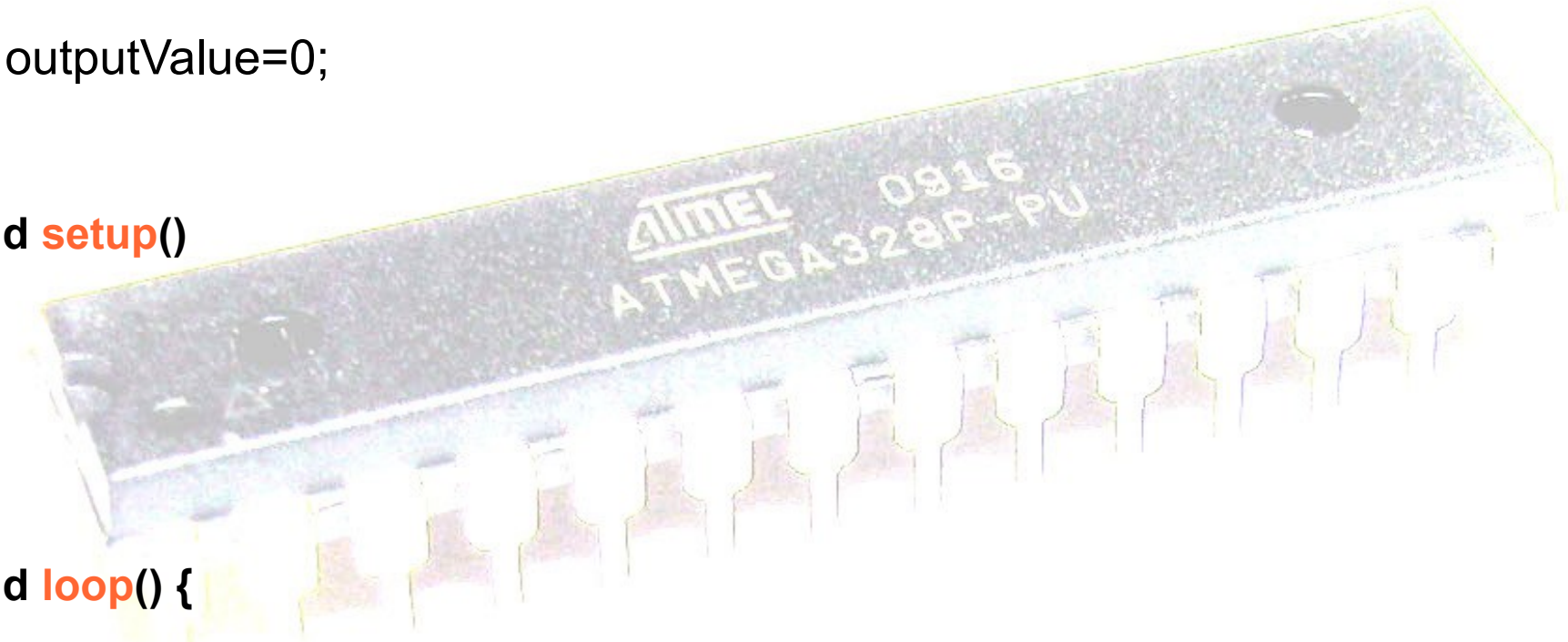
Si vuole controllare la lancetta di uno strumento analogico dallo sketch. Le letture variabili sono più facili da interpretare con strumenti analogici.

Soluzione

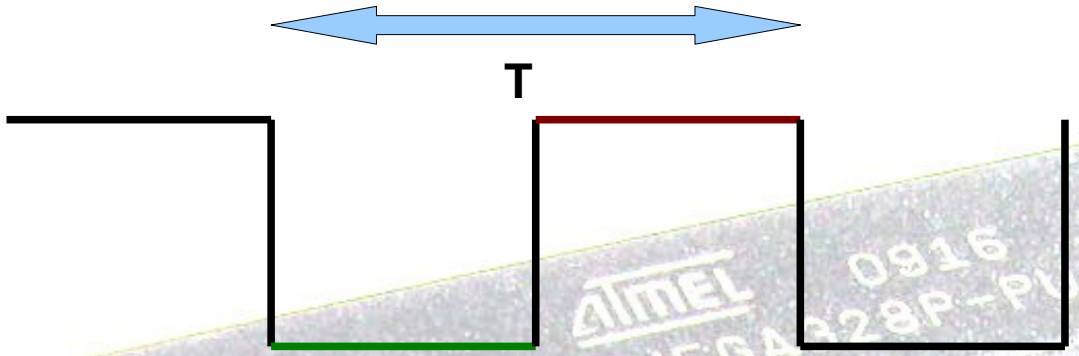
Utilizzo di pwm . Per un misuratore da 1 mA fondo scala si utilizza una resistenza da 5K ohm



```
const in analogPin= 0;
const int analogMeterPin =9;


int sensorValue =0;
int outputValue=0;


void setup()
{
}

void loop() {
   sensorValue= analogRead(analogPin);
   outputValue=map(sensorValue,0,1023,0,255);
   analogWrite(analogMeterPin, outputValue);
}
```



La funzione analogWrite(pin,0-255) utilizza una tecnica detta PWM (Pulse Width Modulation) . Il segnale PWM opera variando la proporzione dei tempi di on e di off degli impulsi.



analogWrite(pin ,127) ; //50% del duty cycle

Il periodo è di 1 ms . Su Arduino i pin che possono essere utilizzati per l'output analogico sono il 3, 5,6, 9, 10 e 11.

SENSORI DI TEMPERATURA

LM35 produce un voltaggio analogico direttamente proporzionale alla temperatura con un output di 10 millivolt per grado.

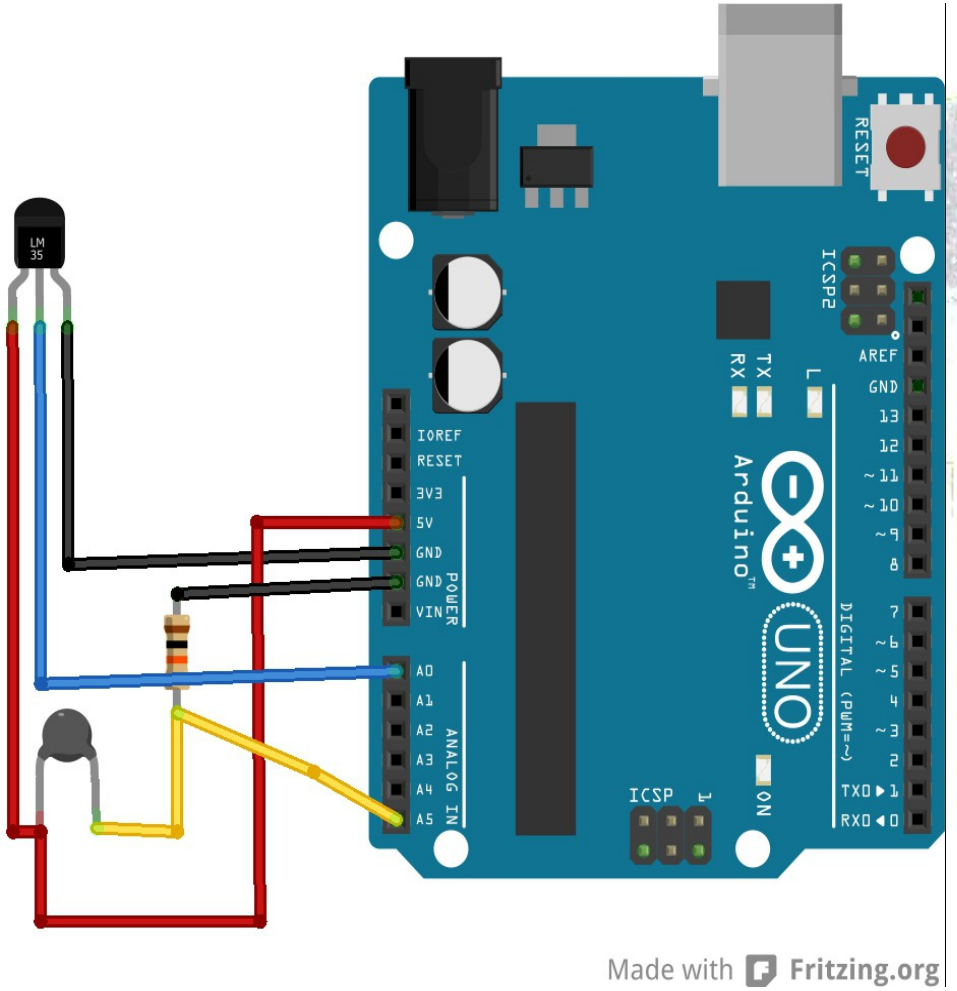
Il termistore NTC da 10 K ohm a 25° varia la sua resistenza con la temperatura.

Con istruzioni di Arduino

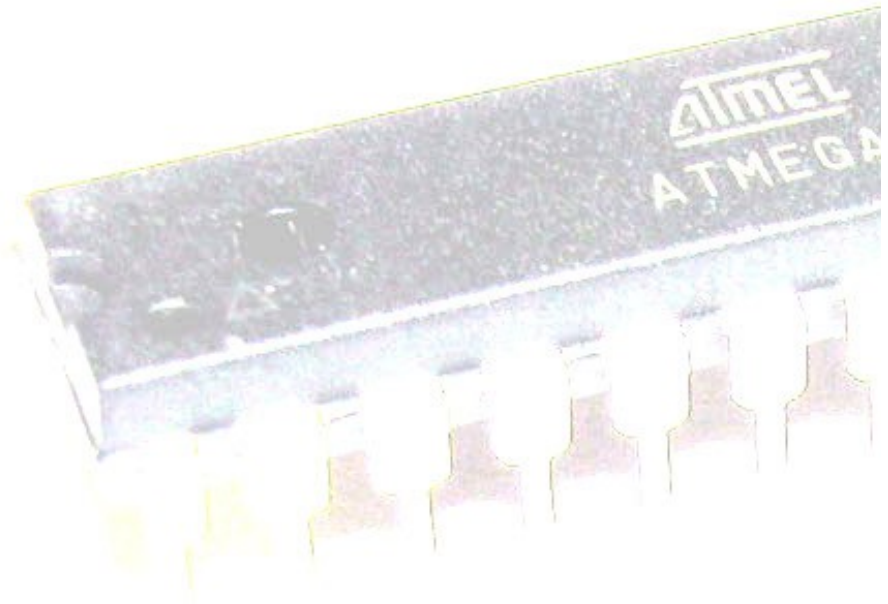
```
voltaggio=(5.0*valoreLetto)/1023;

termistore= ((10000*5.0)/voltaggio)-10000);

temp= (4100.0/log(termistore/0.01066);

// la costante 4100 dipende dal termistore
```

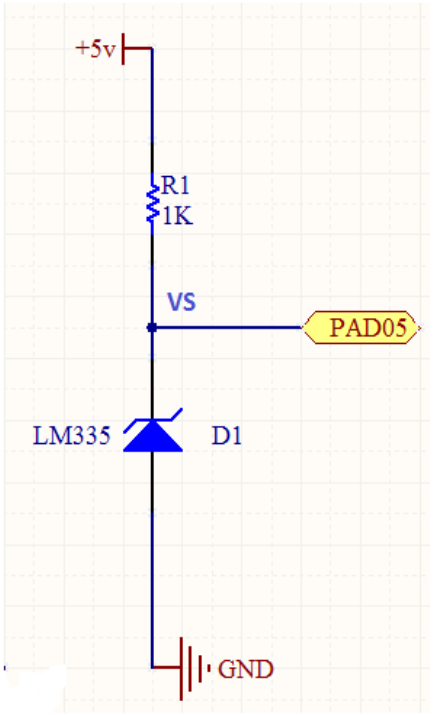
0.01066 = 10000 e^ (-4100/(273.15+25))

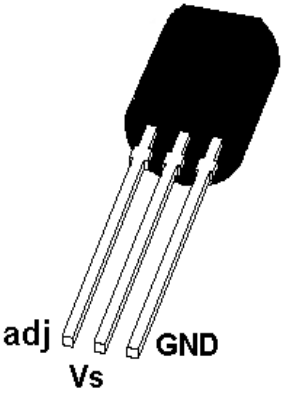
Sensori di temperatura



Un sensore alternativo per misurare la temperatura è l'LM335. L'output dell'LM335 è pari a 10mV per grado Kelvin e quindi 0° celsius producono 2,731 volt. Per impostare la corrente di impiego è necessaria una resistenza da 1 K ohm.





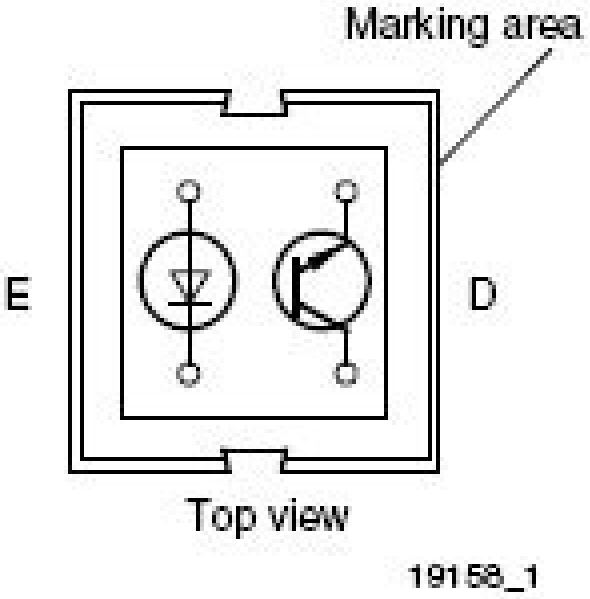



```
int value= analogRead(inPin);
float millivolts= (value/ 1023.0)*5000;
float celsius= (millivolts/10) -273.15;
```

SENSORI DI PROSSIMITA' A INFRAROSSI

21835

21835

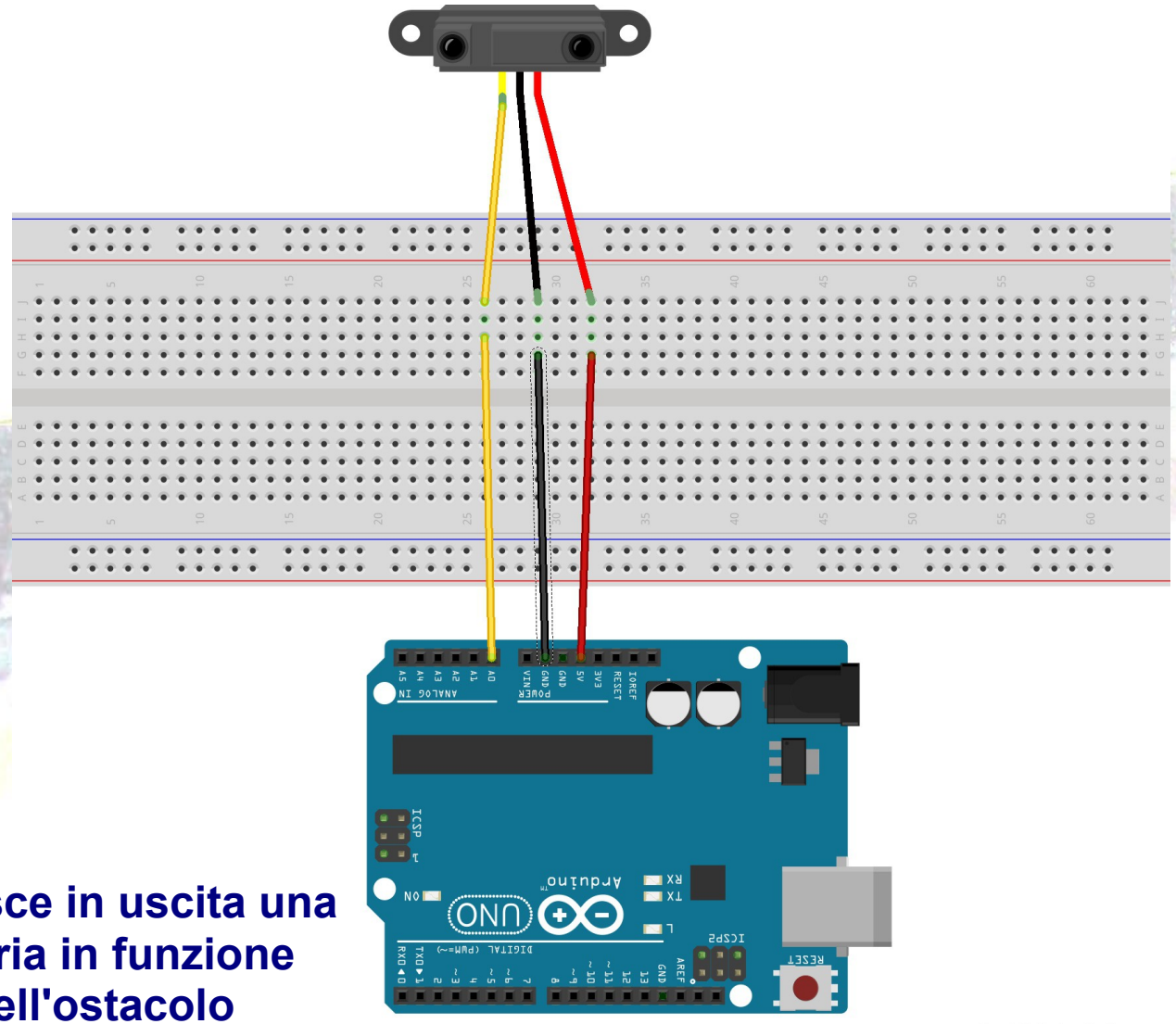


19158_1

DESCRIPTION

The CNY70 is a reflective sensor that includes an infrared emitter and phototransistor in a leaded package which blocks visible light.

SENSORI DI PROSSIMITA' A INFRAROSSI



Il sensore fornisce in uscita una tensione che varia in funzione della distanza dell'ostacolo rilevato. Si utilizza una analogRead(A0)

Rispondere a un telecomando ad infrarossi

Problema

Si vuole rispondere alla pressione di un tasto qualsiasi sul telecomando di un televisore o altro.

Soluzione

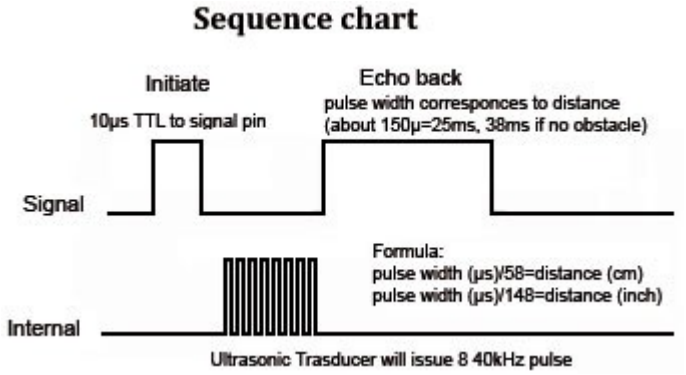
Si utilizza un dispositivo *modulo ricevitore IR* che ha al suo interno un amplificatore e un filtro a 38 Khz per renderlo immune alla radiazione infrarossa ambientale.

```
// il led sul pin 13 si accende o si spegne ogni volta
// che si preme un tasto
#include <IRremote.h>   // libreria da includere
const int RECV_PIN = 11;  // output è collegato al pin 11
const int ledPin = 13;
IRrecv irecv(RECV_PIN);
decode_results results;

void setup()
{
 pinMode(ledPin,OUTPUT);
 irecv.enableIRIn(); // Start oggetto ricevitore
}
 boolean lightState = false;
 unsigned long last= millis();


void loop() {
 if (irecv.decode(&results)== true) {
    if( millis() -last >250) {         // se sono trascorsi almeno 250 ms
      lightState = ! lightState;
      digitalWrite(ledPin, lightState);   // cambia stato
    }
    last=millis();
    irecv.resume(); // Riceve il prossimo valore
 }
}
```

Utilizzo di un misuratore di distanza ad ultrasuoni

Fornendo un impulso positivo della durata di 10 µs al pin di ingresso “Trigger”, il sensore genera un treno d'impulsi (otto) ad ultrasuoni a 40 kHz, questi impulsi vengono inviati, tramite la capsula trasmittente, verso l'ostacolo, la capsula ricevente rileva l'eco sul pin di uscita “Echo”, dove sarà presente un impulso di durata da 150 µs (2 cm) a 25 ms (4,5 m) proporzionale alla distanza dell'oggetto.

Se non rileva alcun ostacolo entro i 4,5 m, produce un impulso di durata di circa 38 ms. La formula per calcolare la distanza in cm è: impulso in µs diviso 58.



```
const int trig=2;    // uscita per inviare un impulso al sonar
const int echo=3; // ingresso per leggere l'eco

void setup() {
  pinMode(trig,OUTPUT);
  pinMode(echo,INPUT);
  Serial.begin(9600);
}

void loop() {
  Serial.print(distanza());
  Serial.println("  cm");
  delay(2000);
}
int distanza(){
  long duration;
  digitalWrite(trig,LOW);
  delayMicroseconds(5);
  digitalWrite(trig,HIGH);
  delayMicroseconds(10);  // creazione di un impulso di 10 us
  digitalWrite(trig,LOW);
  duration=pulseIn(echo,HIGH,20000); // rileva il tempo di ritorno dell'eco
  return (int) duration/58;  // velocità del suono 340 m/s  0,034 cm/us   29 us/cm
}
```

Timeout in microsecondi
opzionale

L'output fisico

Controllare un motore in corrente continua

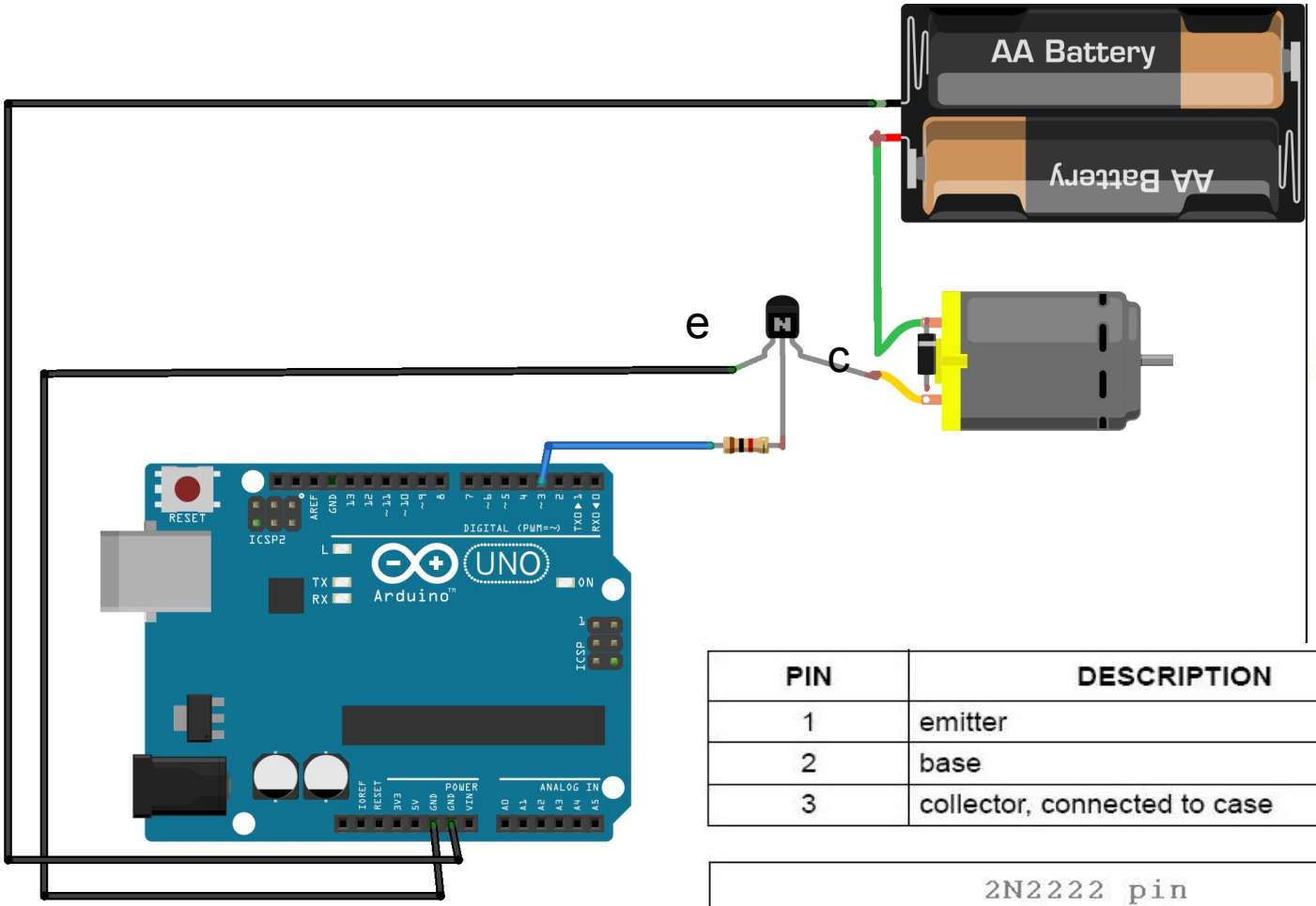
Problema

Si vuole controllare la velocità di rotazione di un motore in c.c. attraverso una uscita PWM (Pulse Wide Modulation)

Soluzione

Si utilizza la funzione analogWrite(pin, 0-255) per regolare il pwm.

Lo sketch seguente riceve un carattere da 0 a 9 dalla porta seriale e lo converte nel range da 0 a 255 per comandare il motore.



PIN	DESCRIPTION
1	emitter
2	base
3	collector, connected to case

2N2222 pin configuration

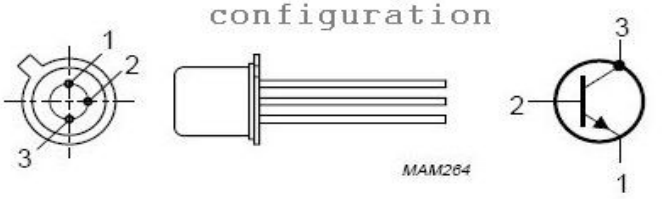


Fig.1 Simplified outline (TO-18) and symbol.

```
const int motorPin = 3;

void setup() {
   Serial.begin(9600);
}

void loop() {
   if (Serial.available()) {
      char ch= Serial.Read();
      if(isDigit(ch)       // ch è un numero ?
{
         int speed=map(ch,'0','9',0,255);
         analogWrite(motorPin,speed);
         Serial.println(speed);
      }
      else
      {
         Serial.println(“Carattere inaspettato ”);
         Serial.println(ch);
      }
   }
}
```

Controllare la rotazione di due motori in c.c. con un ponte ad H.

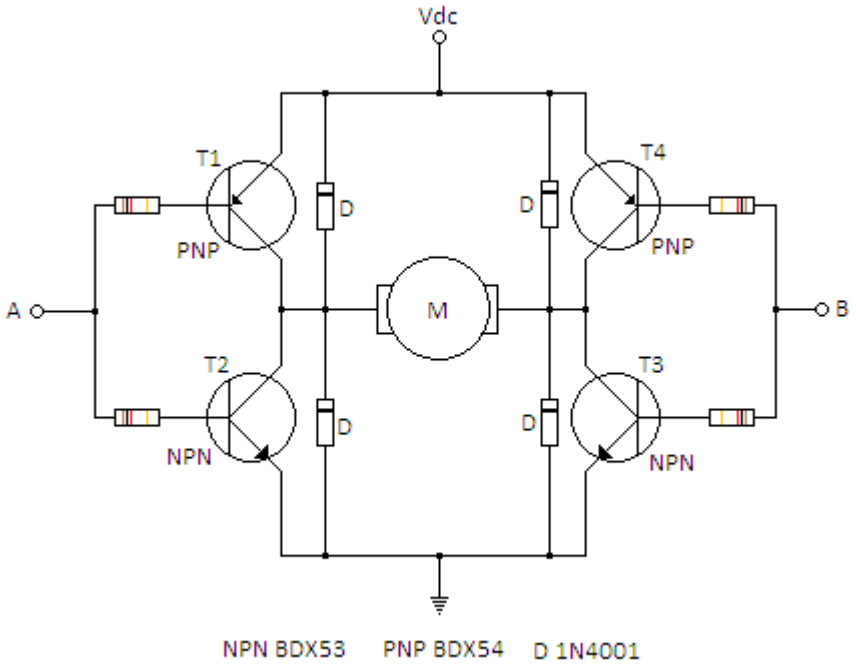
Problema

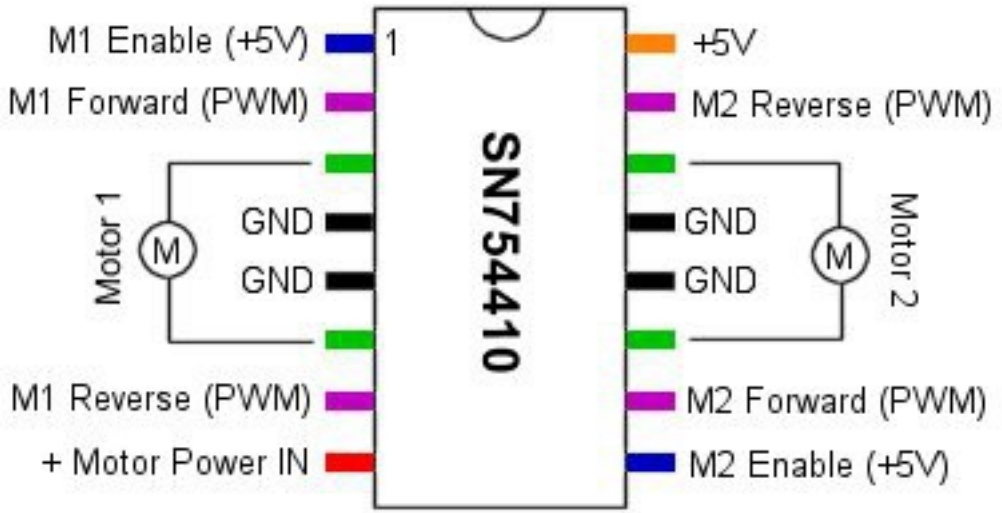
Si vuole controllare la direzione di rotazione di due motori in c.c. Tale problema è tipico del controllo dei motori di un robot autonomo.

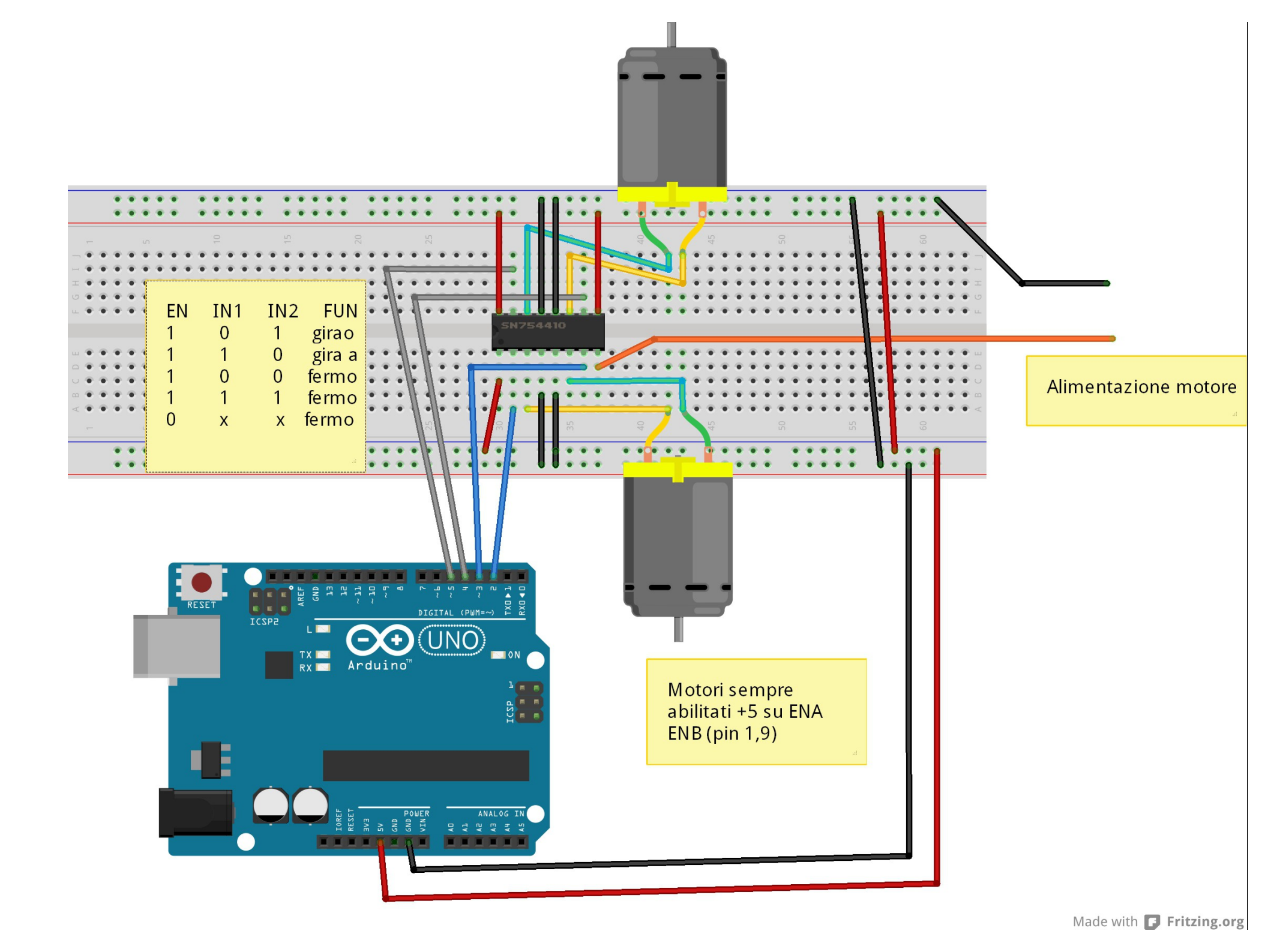
Soluzione

I ponti ad H sono in grado di controllare i motori a spazzole. Si può utilizzare un L293D H-Bridge IC che può controllare due motori in c.c.

Si può utilizzare anche l'SN754410, che presenta la stessa disposizione dei pin.






EN IN1 IN2 FUN
1 0 1 girao
1 1 0 gira a
1 0 0 fermo
1 1 1 fermo
0 x x fermo
SN754410
Alimentazione motore
Motori sempre abilitati +5 su ENA ENB (pin 1,9)
UNO
Arduino
Made with Fritzing.org

```
const int m1Pin1 =5;
const int m1Pin2 =4;
const int m2Pin1 =3;
const int m2Pin2 =2;

void setup() {
  pinMode(m1Pin1,OUTPUT);
  pinMode(m1Pin2,OUTPUT);
  pinMode(m2Pin1,OUTPUT);
  pinMode(m2Pin2,OUTPUT);
}

void loop() {
  avanti(2000);
  fermo();
  indietro(2000);
}

void avanti(int tempo) {
  digitalWrite(m1Pin1,LOW);
  digitalWrite(m1Pin2,HIGH);
  digitalWrite(m2Pin1,LOW);
  digitalWrite(m2Pin2,HIGH);
  delay(tempo);

}
```

```
void indietro(int tempo) {
  digitalWrite(m1Pin1,HIGH);
  digitalWrite(m1Pin2,LOW);
  digitalWrite(m2Pin1,HIGH);
  digitalWrite(m2Pin2,LOW);
  delay(tempo);
}

void fermo() {
  digitalWrite(m1Pin1,LOW);
  digitalWrite(m1Pin2,LOW);
  digitalWrite(m2Pin1,LOW);
  digitalWrite(m2Pin2,LOW);
  delay(100);

}
```

Controllare la posizione di un servomotore

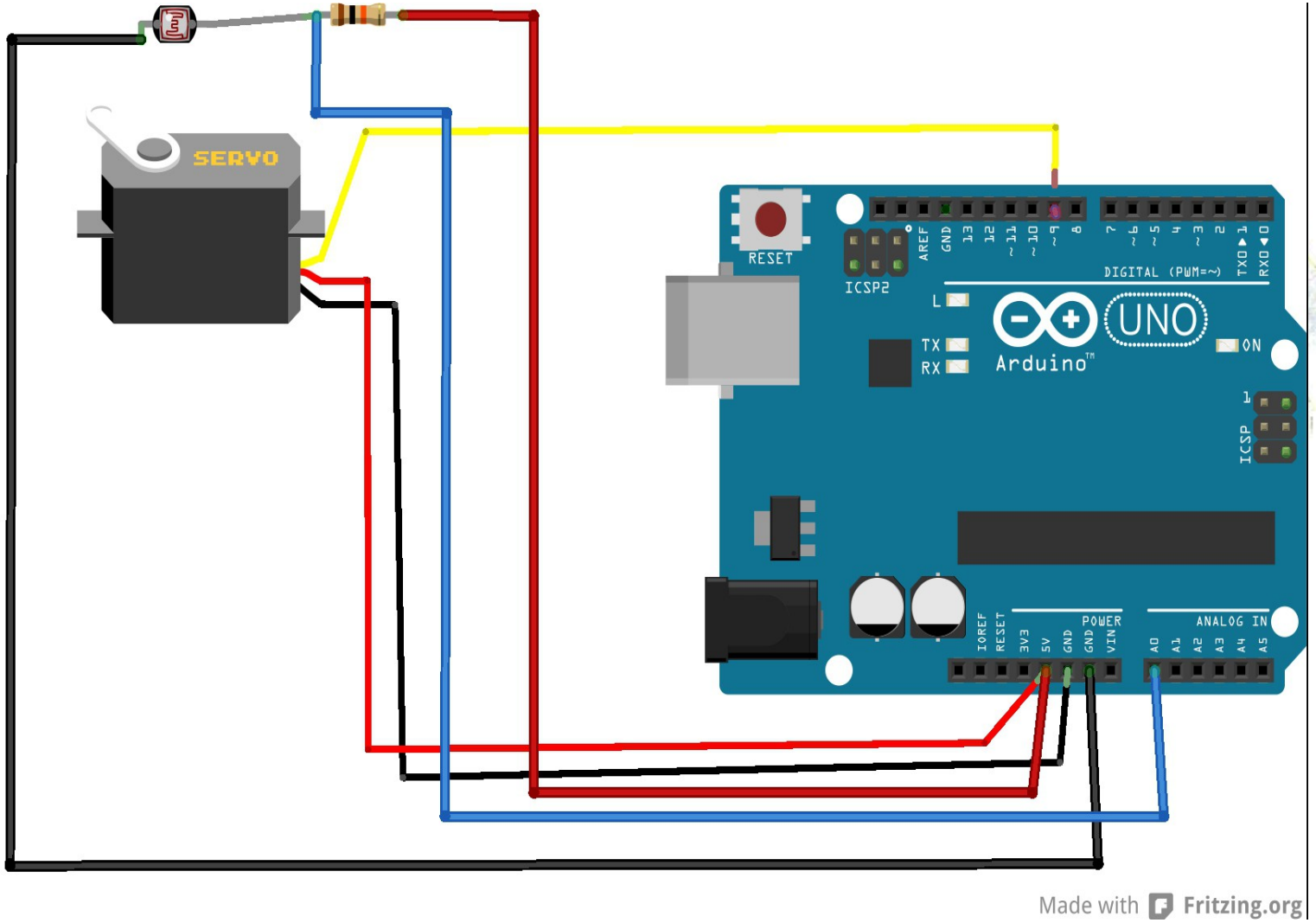
Problema

Si vuole controllare la posizione di un servomotore posizionandolo da 0° a 180°. Utilizzando un sensore di luce che ruota con il servomotore si vuole che questo si sposti nella posizione di masssima luce.

Soluzione

Si utilizza una fotoresistenza fissata sul servomotore, si fa ruotare il servomotore da 0° a 180° a passi di un grado e si aggiorna man mano il valore di luce massimo misurato dalla fotoresistenza aggiornando nel contempo l'angolo corrispondente. Alla fine si riporta il servomotore nella posizione corrispondente all'angolo di luce massima.

Il servomotore si posiziona a seconda della durata degli impulsi che riveve: 1 ms corrisponde a 0°, 2 ms a 180°.
La durata degli impulsi dipende dal tipo di servomotore.


SERVO
RESET
ICSP2
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TXD▲1
RXD◀0
DIGITAL (PWM=~)
L
TX
RX
UNO
Arduino™
ON
ICSP
IOREF
RESET
3V3
5V
GND
GND
VIN
POWER
A0
A1
A2
A3
A4
A5
ANALOG IN
Made with Fritzing.org

```
#include <Servo.h>

Servo servox;
int letturamin=1024; // la luce fa diminuire il valore della fotoresistenza e la lettura
int angolomax=0;

void setup() {
  servox.attach(9,1600,1900);  //tra 1,6 ms e 2 ms  (usare pin 9 o 10)
  servox.write(90);
  delay(100);
}

void loop() {
  for(int angolo=20; angolo<=180;angolo=angolo++)
  {
  servox.write(angolo);
  delay(30);
  int lettura=analogRead(A0);
  if (letturamin> lettura)
     {
       letturamin=lettura;
       angolomax=angolo;
     }
  }
                                                    servox.write(angolomax);
                                                      delay(20);
                                                      angolomax=0;
                                                      letturamin=1024;

                                                      delay(5000);
                                                    }
```

Controllare la rotazione di un motore passo passo (stepper motor)

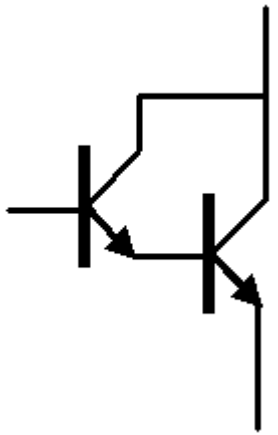
Problema

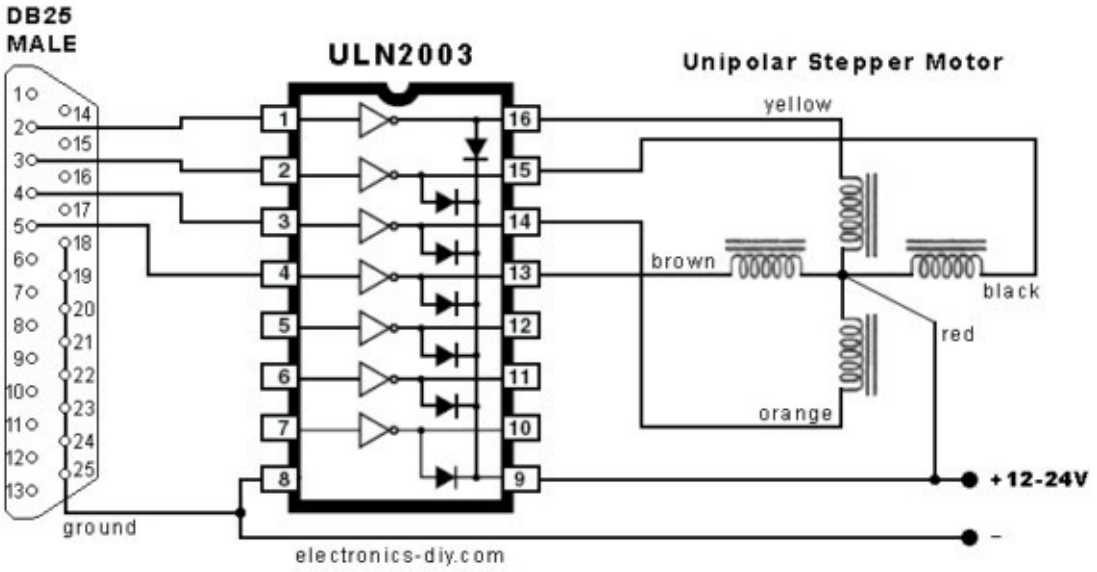
Si vuole controllare la rotazione di un motore passo passo unipolare utilizzando gli output digitali.

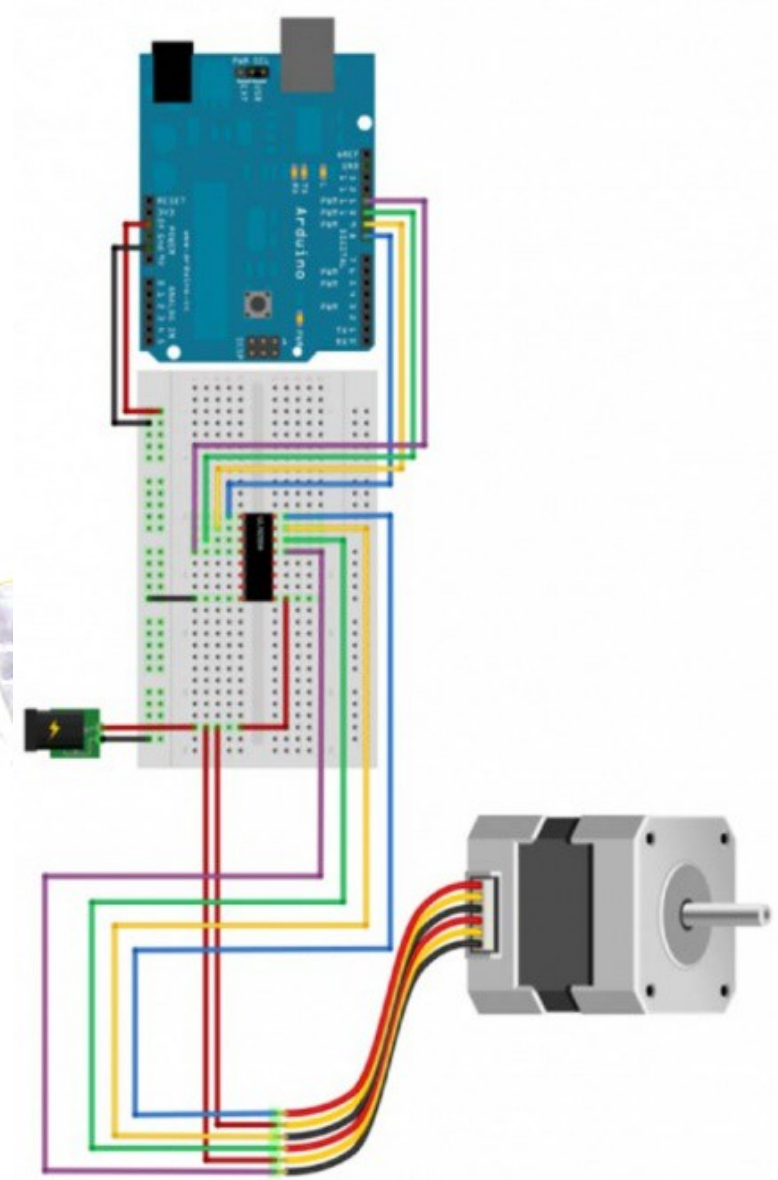
I motori passo passo si distinguono in due categorie, bipolari e unipolari . Quelli unipolari prevedono 4 avvolgimenti da alimentare in sequenza uno alla volta in senso orario o antiorario per generare la rotazione. Sono caratterizzati da 5 o 6 fili a seconda che gli avvolgimenti siano collegati ad un nodo comune o a due a due con due nodi.

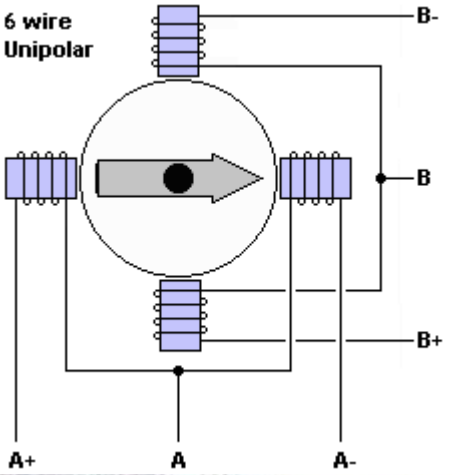
Soluzione

Ogni volta che si alimenta un avvolgimento dello statore, togliendo alimentazione agli altri, il motore gira di un passo, per esempio 1,8° in quanto il rotore, magnetizzato nord sud, si orienta sul campo magnetico . Per alimentare i 4 avvolgimenti dello statore si può utilizzare un darlington arrays come un ULN2003.

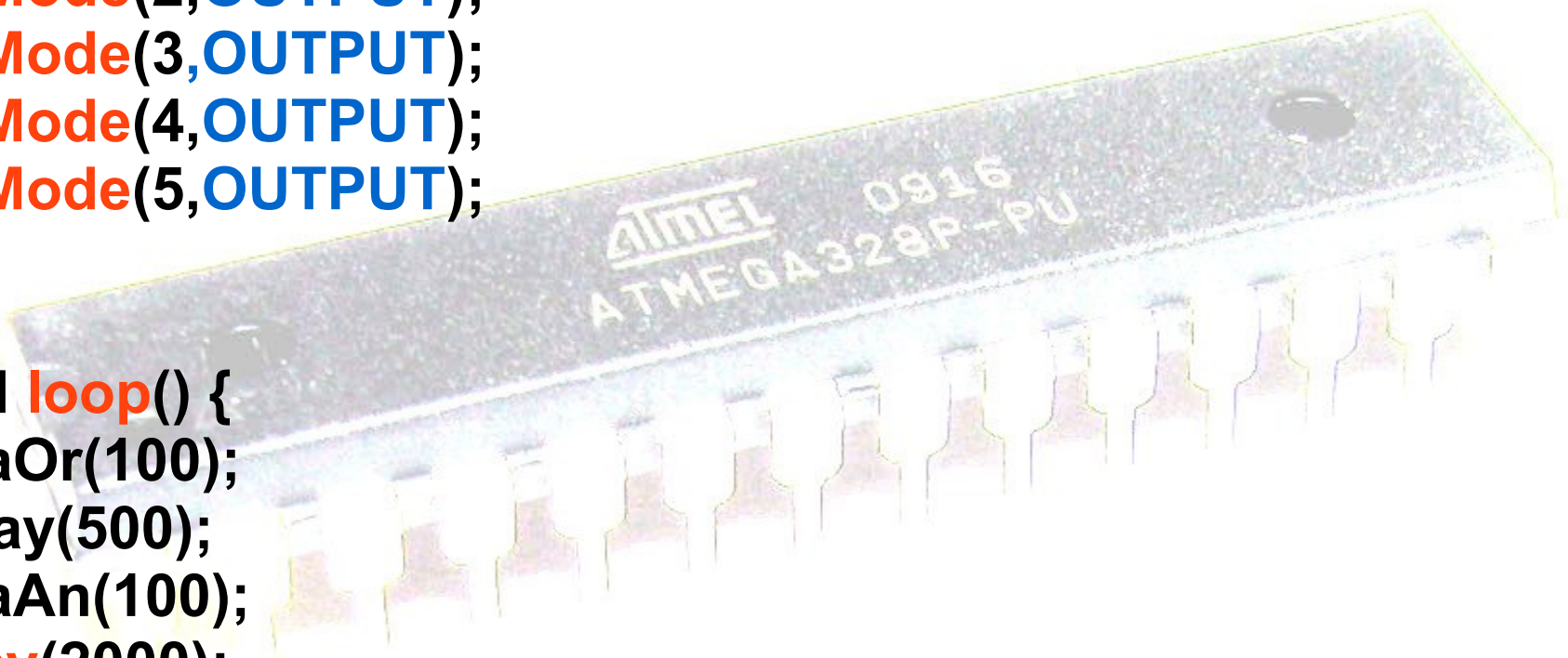







6 wire
Unipolar
B-
B
B+
A+
A
A-

```
const int fase[4]= {0b0001,0b0010,0b0100,0b1000};
const int pin[4]= {2,3,4,5};

void setup() {
pinMode(2,OUTPUT);
pinMode(3,OUTPUT);
pinMode(4,OUTPUT);
pinMode(5,OUTPUT);
}

void loop() {
 giraOr(100);
 delay(500);
 giraAn(100);
delay(2000);
}
```



```
void giraOr(int numPassi) {  // numPassix4
 for(int passi=0;passi< numPassi;passi++) {
  for (int j=0;j<4;j++) {   // per le 4 fasi
   for(int k=0;k<4;k++){   // per i 4 bit
     digitalWrite(pin[k],bitRead(fase[j],k));
    }
   delay(3); // ritardo di passo
  }
}
for(int k=0;k<4;k++)   // togli alimentazione
  digitalWrite(pin[k],LOW);
}


void giraAn(int numPassi) {   // numPassix4
 for(int passi=0;passi< numPassi;passi++) {
  for (int j=3;j>=0;j--) {
   for(int k=0;k<4;k++){
     digitalWrite(pin[k],bitRead(fase[j],k));
   }
  delay(3);
  }
}
for(int k=0;k<4;k++)  // togli alimentazione
  digitalWrite(pin[k],LOW);
}
```

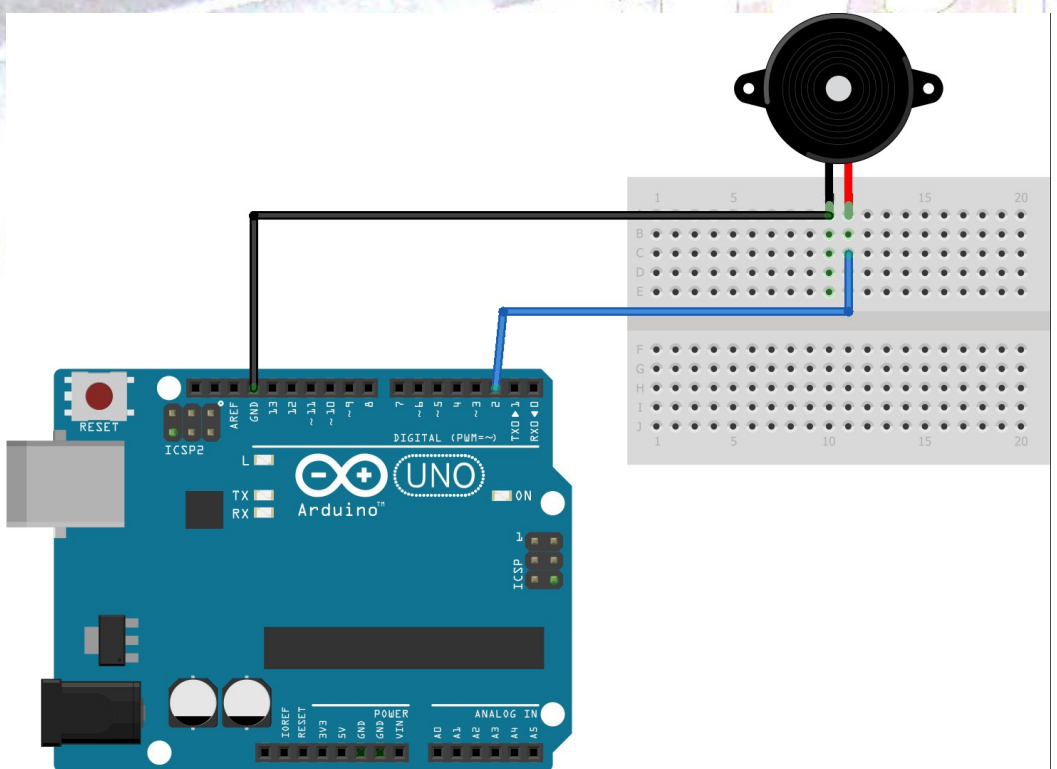
L'output audio

Un suono è prodotto dalla vibrazione dell'aria. Il segnale elettrico viene convertito in segnale acustico, vibrazioni dell'aria, da un altoparlante o da un dispositivo piezoelettrico.

L'intervallo dei suoni udibili dagli esseri umani oscilla tra 20 Hz (cicli o periodi al secondo) e 20000 Hz.

Il software di Arduino per produrre del suono include la funzione tone

tone(pin, frequenza in Hz, durata in ms)



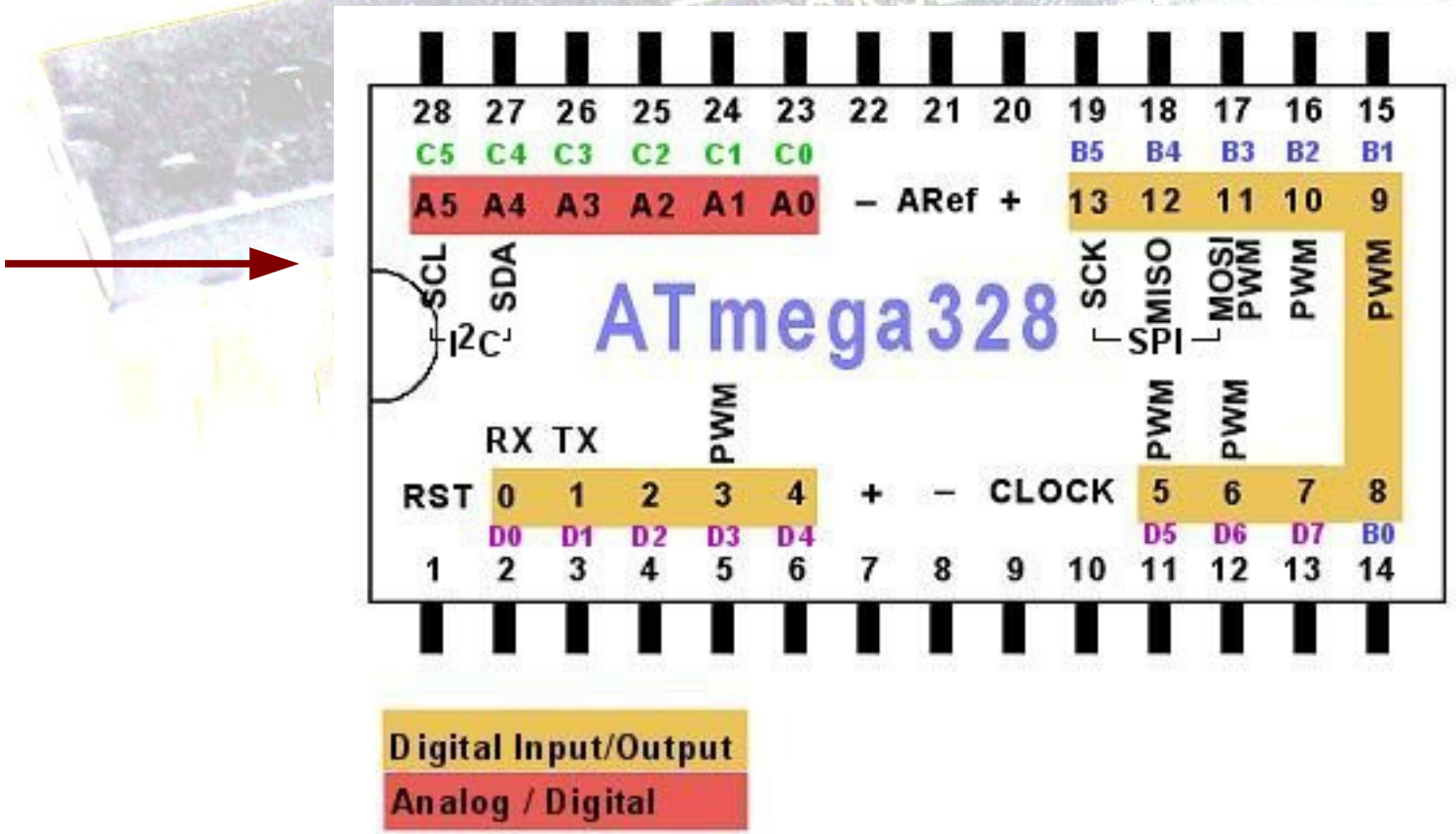
```
const int uscita =2;
const int la4=440;       // do re mi fa sol la si do   12 semitoni, 1 tra mi-fa e si-do
const int si4=440*pow(2,(2.0/12));  // 440 Herz  x  2^(semitoni/12)
const int do5 =440*pow(2,(3.0/12));
const int re5=440*pow(2,(5.0/12));
const int sib4=440*pow(2,(1.0/12));
const int sol4=440*pow(2,-(2.0/12));

const int
tuScendi[]={do5,do5,re5,do5,sib4,sib4,la4,la4,sol4,la4,sib4,do5,do5,sib4,la4,sol4};

void setup() {
  pinMode(uscita,OUTPUT);
}

void loop() {
  for(int k=0;k<sizeof(tuScendi)/sizeof(int);k++){
 tone(uscita,tuScendi[k],750);

 delay(750);
  }
 delay(3000);
}
```

(pin, frequenza in Hz, durata in ms)

Comunicare utilizzando I2C

Introduzione

I due collegamenti del bus I2C si chiamano SCL e SDA. Sulle schede Arduino standard sono disponibili utilizzando il pin analogico 5 per SCL, che offre il segnale di clock, e il pin analogico 4 per SDA che serve per trasferire i dati. Un dispositivo sul bus I2C funziona da dispositivo *master* . La sua funzione è coordinare il trasferimento delle informazioni con gli altri dispositivi *slave.*



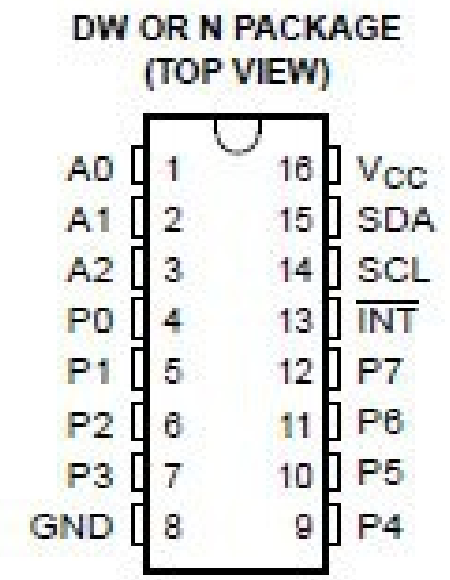
Utilizzare un port expander I2C

Problema

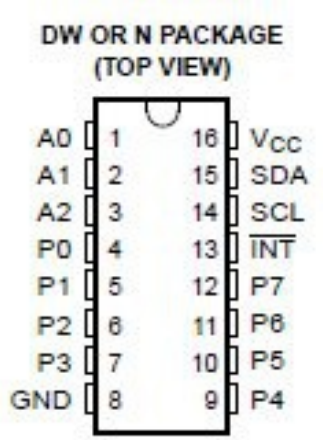
Si vogliono utilizzare più porte di quante me fornisca la propria scheda.

Soluzione

Si utilizza un port expander esterno, come PCF8574A, che è dotato di 8 pin di input/output che possono essere controllati usando I2C.



Gli ingressi A0-A2 servono per stabilire l'indirizzo del dispositivo. Collegandoli a massa l'indirizzo è 0x38 e arriva fino a 0x3F.



La libreria da includere è Wire.h e i medodi sono:

```
Wire.begin();
Wire.beginTransmission(address);
Wire.write(numero);
Wire.endTransmission();
```

```
#include <Wire.h>
const int address = 0x38;
int numero;

void setup()
{
  Wire.begin(); }

void loop() {
  for( numero=0; numero <256; numero++) {
    Wire.beginTransmission(address);
    Wire.write(numero ^ 0xFF);   // si accendono dei led in logica negata
    Wire.endTransmission();
   delay(200);
  }
  }
```

Utilizzare l'hardware del controller

Gli interrupt

Gli interrupt sono segnali che obbligano il processore ad interrompere il flusso normale di uno sketch per gestire una operazione che richiede un'attenzione immediata, prima di continuare quello che stava facendo.

Arduino utilizza gli interrupt per gestire i dati che provengono dalla porta seriale, per tenere il tempo nelle funzioni delay e millis e per lanciare una funzione con attachInterrupt.

Arduino gestisce un interrupt alla volta. Il codice che gestisce un interrupt, detto *interrupt service routine*, dovrebbe essere breve, per evitare che gli altri interrupt vengano gestiti con un ritardo eccessivo.

La funzione attachInterrupt(0,analyze,CHANGE) collega all'interrupt 0, che si riferisce al pin 2, la funzione analyze che viene eseguita quando sul pin 2 avviene un cambiamento di stato. Altre possibilità sono LOW, RISING, FALLING. L'interrupt 1 si riferisce al pin 3 con le stesse modalità.

Utilizzare gli interrupt sui pin 2 e 3

Problema

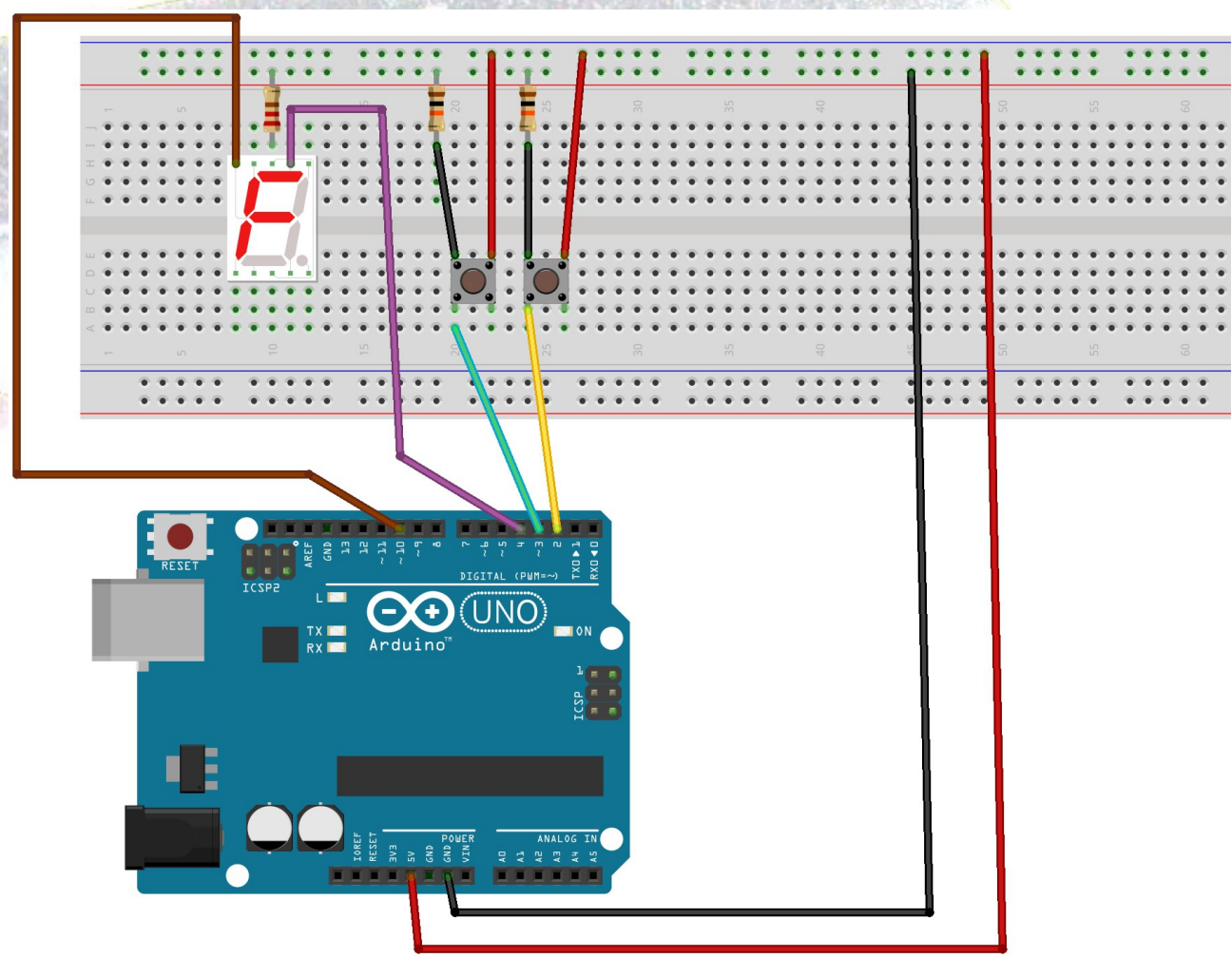
Si vuole realizzare un timer che decrementa ogni 10 secondi un numero visualizzato su di un display. Due pulsanti permettono di incrementare o decrementare tale numero in modo immediato.

Soluzione

Per poter reagire in modo immediato i due pulsanti sono collegati ai pin 2 e 3 che sono sorgenti di interrupt.

```
const int segmentPins[7]={4,5,6,7,8,9,10};
const int attivazione[10]={0b00111111,0b00000110,0b01011011,
0b01001111,0b01100100,0b01101101,0b01111101,0b00000111,0b01111111,0b01101111};
int numero=0;

void showDigit(int number) {
  boolean isBitSet;
  // abilitazione digit
  for (int segment=0; segment<7; segment++)
  {
    isBitSet=bitRead(attivazione[number],segment);
    digitalWrite(segmentPins[segment],isBitSet);
  }
}

void incrementa() {
  // debounce
  if (numero < 9) numero++;
     showDigit(numero);
  }

void decrementa() {
  // debounce
  if(numero > 0) numero--;
     showDigit(numero);
 }
```

```
void setup() {
 for (int i=0;i<8;i++)
   pinMode(segmentPins[i],OUTPUT);
   attachInterrupt(0,incrementa,RISING);
   attachInterrupt(1,decrementa,RISING);
}

void loop() {
  showDigit(numero);
  delay(10000);
  if (numero>0) numero-- ;
}
```



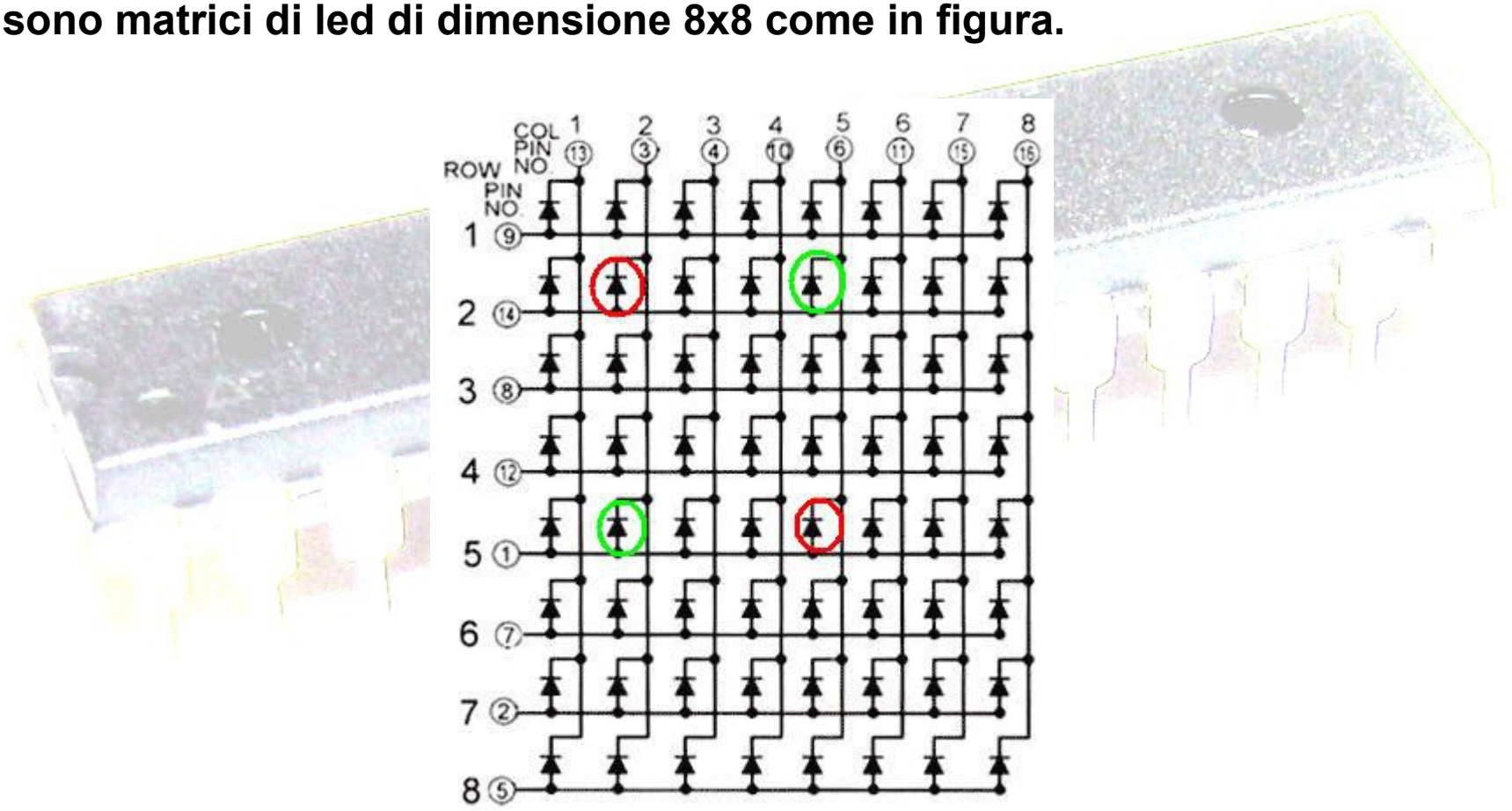
Interrupt dal timer1

```
#include <TimerOne.h>    // è necessario aggiungere la libreria TimerOne
 void setup()
{
  // Initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards
  pinMode(13, OUTPUT);

  Timer1.initialize(100000); // set a timer of length 100000 microseconds (or 0.1 sec - or
10Hz => the led will blink 5 times, 5 cycles of on-and-off, per second)
  Timer1.attachInterrupt( timerIsr ); // attach the service routine here
}

void loop()
{
  // Main code loop
}


//  ISR Timer Routine

void timerIsr()
{
    // Toggle LED
    digitalWrite( 13, digitalRead( 13 ) ^ 1 );
}
```

Controllare una matrice di led utilizzando il multiplexing e l'interrupt del timer

Soluzione

Nello sketch si utilizza una matrice di led composta da 35 led (7 righe x 5 colonne). Ci sono matrici di led di dimensione 8x8 come in figura.



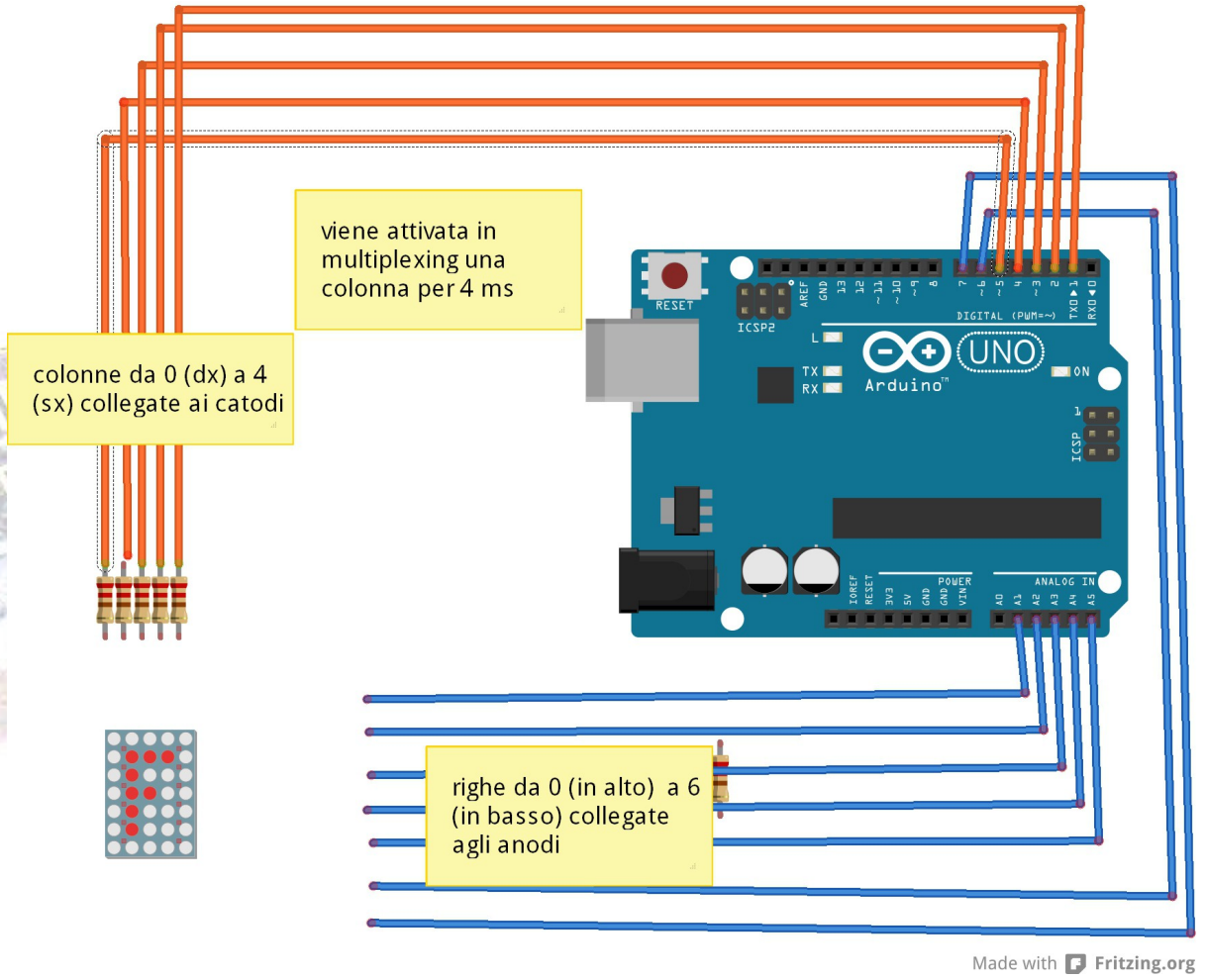
Gli anodi sono collegati alle righe, i catodi alle colonne. Per accendere un led la colonna corrispondente deve essere collegata a massa con resistenza da 220 ohm mentre viene contemporaneamente attivata la riga.

Sono previste cinque diverse schermate che verranno visualizzate in sequenza. La libreria TimerOne permette di utilizzare gli interrupt del timer1.

```
#include <TimerOne.h>
// non può essere attivala la seriale ! per via del pin 1
const int columnPins[]={2,3,4,5,1};
const int rowPins[]= {15,16,17,18,19,6,7}; // A1 A2 A3 A4 A5 6 7  quattro pin di norma analogici sono utilizzati come digitali
int schermo[7][5];
const int ritardo=200;
const int schermata0[7][5]={ {0,0,0,0,0},{0,0,0,0,0},{0,0,0,0,0},{0,0,0,0,0},{0,0,0,0,0},{0,0,0,0,0},{0,0,0,0,0}};
const int schermata1[7][5]={ {0,0,0,0,0},{0,0,0,0,0},{0,0,0,0,0},{0,0,1,0,0},{0,0,0,0,0},{0,0,0,0,0},{0,0,0,0,0}};
const int schermata2[7][5]={ {0,0,0,0,0},{0,0,0,0,0},{0,1,1,1,0},{0,1,0,1,0},{0,1,1,1,0},{0,0,0,0,0},{0,0,0,0,0}};
const int schermata3[7][5]={ {0,0,0,0,0},{0,1,1,1,0},{0,1,0,1,0},{0,1,0,1,0},{0,1,0,1,0},{0,1,1,1,0},{0,0,0,0,0}};
const int schermata4[7][5]={ {1,1,1,1,1},{1,0,0,0,1},{1,0,0,0,1},{1,0,0,0,1},{1,0,0,0,1},{1,0,0,0,1},{1,1,1,1,1}};
int colonna=0;
int colonnaPrec=0;
```

```
void setup() {

  for(int i=0;i<7;i++)
     pinMode(rowPins[i],OUTPUT);
  for(int i=0;i<5;i++){
     pinMode(columnPins[i],OUTPUT);
     digitalWrite(columnPins[i],HIGH);
  }
  Timer1.initialize(4000); // ogni 4 ms
  Timer1.attachInterrupt(visCol);


}

void visCol(){    // richiamata ogni 4 ms dall'interupt del timer
  digitalWrite(columnPins[colonnaPrec],HIGH);   // disattiva la precedente colonna
  for(int k=0;k<7;k++)
    digitalWrite(rowPins[k],schermo[k][colonna]);    // invio dati colonna
   digitalWrite(columnPins[colonna],LOW); // attiva la colonna
   colonnaPrec=colonna;
   colonna=((colonna+1) % 5);


}
```

```
void loop() {
  for(int j=0;j<7;j++)
    for(int k=0;k<5;k++)
      schermo[j][k]=schermata0[j][k];
    delay(ritardo);   // 200 ms
    for(int j=0;j<7;j++)
    for(int k=0;k<5;k++)
      schermo[j][k]=schermata1[j][k];
    delay(ritardo);
     for(int j=0;j<7;j++)
    for(int k=0;k<5;k++)
      schermo[j][k]=schermata2[j][k];
    delay(ritardo);
     for(int j=0;j<7;j++)
    for(int k=0;k<5;k++)
      schermo[j][k]=schermata3[j][k];
    delay(ritardo);
     for(int j=0;j<7;j++)
    for(int k=0;k<5;k++)
      schermo[j][k]=schermata4[j][k];
    delay(ritardo);

  }
```



Impostare rapidamente i pin digitali

Problema

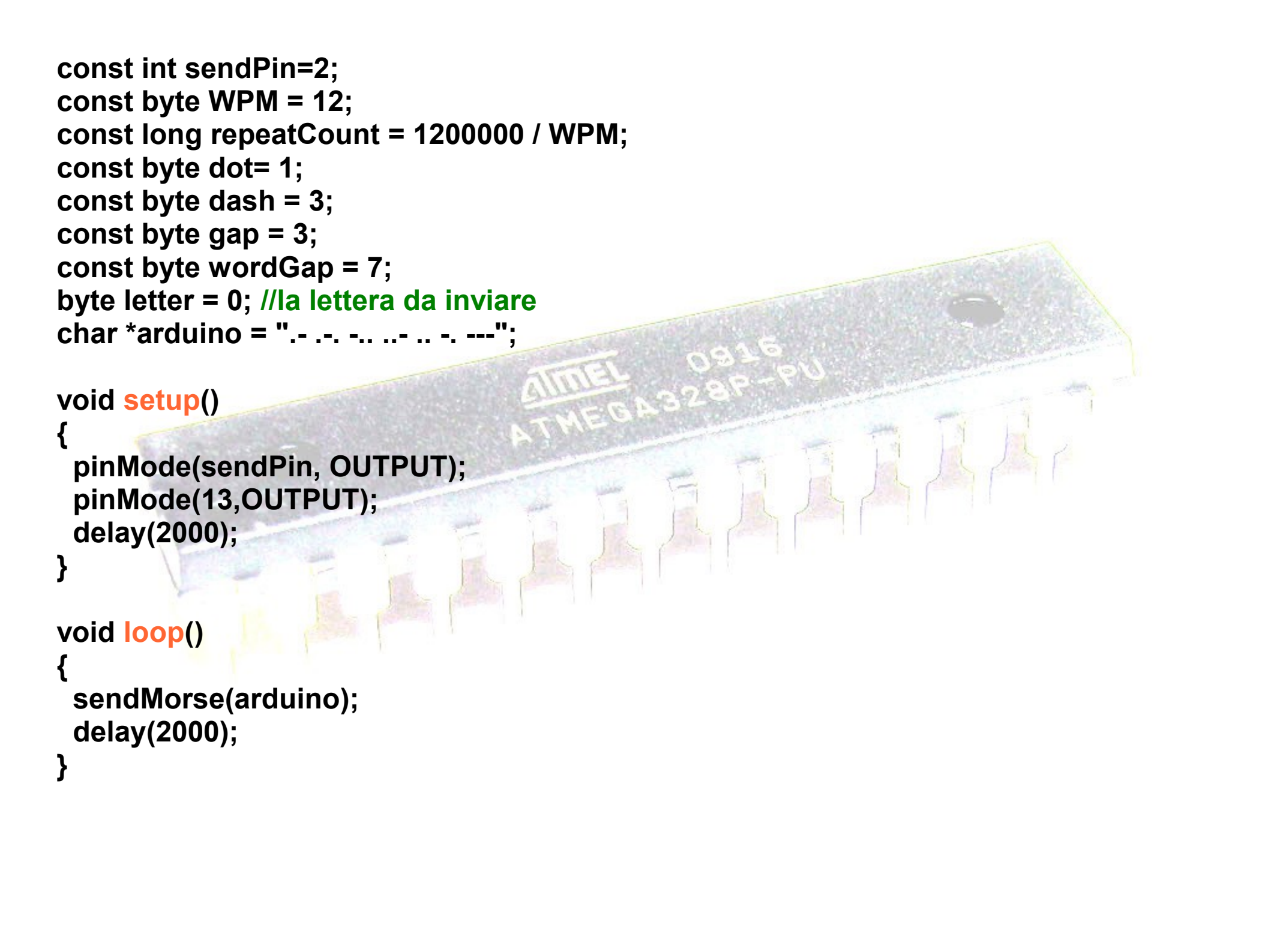
Gestire i pin digitali più velocemente di quanto non permetta la funzione digitalWrite.

Soluzione

La funzione digitalWrite è comoda da utilizzare ma per avere maggiore velocità si possono impostare direttamente i bit sui registri hardware che controllano i pin digitali.

Nello sketch seguente si utilizza l'IO hardware diretto per inviare un codice Morse ad una radio AM sintonizzata su 1 Mhz . La tecnica è 30 volte più veloce della funzione digitalWrite.

```
const int sendPin=2;
const byte WPM = 12;
const long repeatCount = 1200000 / WPM;
const byte dot= 1;
const byte dash = 3;
const byte gap = 3;
const byte wordGap = 7;
byte letter = 0; //la lettera da inviare
char *arduino = ".- .-. -.. ..- .. -. ---";

void setup()
{
  pinMode(sendPin, OUTPUT);
  pinMode(13,OUTPUT);
  delay(2000);
}

void loop()
{
  sendMorse(arduino);
  delay(2000);
}
```



```
void sendMorse(char * string)
{
  letter=0;
  while(string[letter]!=0)
  {
    if (string[letter] == '.')
    sendDot();
    else if(string[letter] == '-')
    sendDash();
    else if (string[letter] == ' ')
    sendGap();
    else if (string[letter] == 0)
    sendWordGap();

    letter=letter+1;
  }
}
```

```
void sendDot()
{ digitalWrite(13,HIGH);
  transmitCarrier(dot * repeatCount);
  digitalWrite(13,LOW);
  sendGap();
}

void sendDash()
{ digitalWrite(13,HIGH);
  transmitCarrier( dash * repeatCount);
  digitalWrite(13,LOW);
  sendGap();
}

void sendGap()
{
  transmitNoCarrier(gap * repeatCount);
}
```

```
void  sendWordGap()
{
  transmitNoCarrier(wordGap *
repeatCount);
}

void  transmitCarrier(long
count)
{
  while (count--)
  {
    bitSet(PORTD,sendPin);
    bitSet(PORTD,sendPin);
    bitSet(PORTD,sendPin);
    bitSet(PORTD,sendPin);
    bitClear(PORTD, sendPin);
  }
}
```

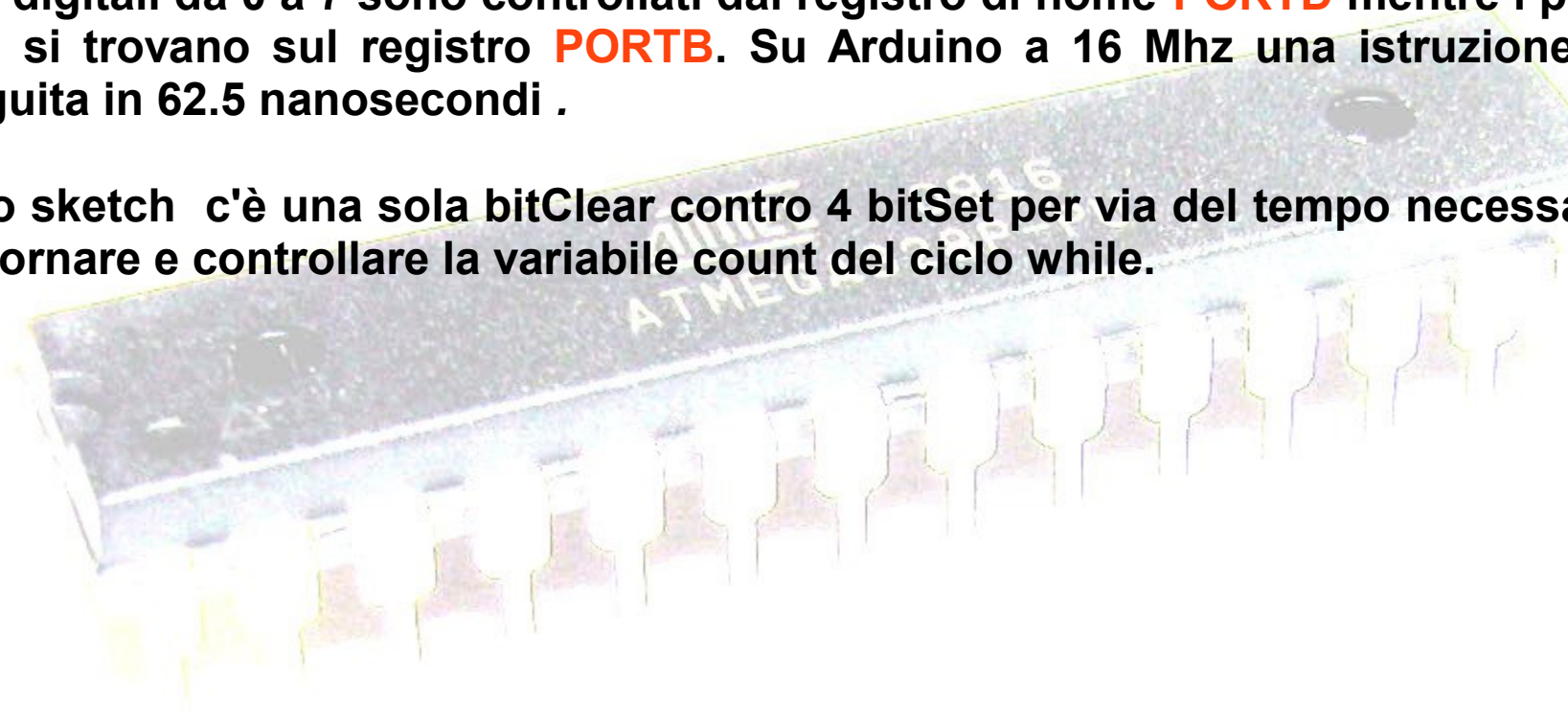
```
void  transmitNoCarrier(long count)
{
  while(count--)
  {
    bitClear(PORTD, sendPin);
    bitClear(PORTD, sendPin);
    bitClear(PORTD, sendPin);
    bitClear(PORTD, sendPin);
    bitClear(PORTD, sendPin);
  }
}
```

Discussione

bitSet e bitClear non sono funzioni ma sono *macro di una sola istruzione in codice macchina.*

I pin digitali da 0 a 7 sono controllati dal registro di nome PORTD mentre i pin da 8 a 13 si trovano sul registro PORTB. Su Arduino a 16 Mhz una istruzione viene eseguita in 62.5 nanosecondi .

Nello sketch c'è una sola bitClear contro 4 bitSet per via del tempo necessario ad aggiornare e controllare la variabile count del ciclo while.



Utilizzare lo shield RTC

Problema

Avere un orologio di sistema

Soluzione

Utilizzare un RTC(Real time clock) shield che permette di avere un orario preciso sollevando il microcontrollore dall'onere di gestire l'orologio di sistema.

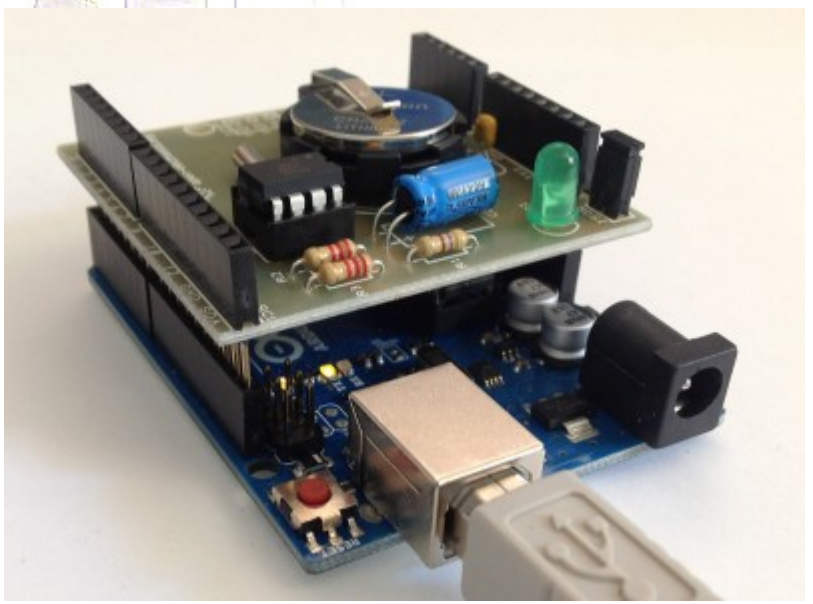
Lo shield viene inserito in una struttura a sandwich con sotto arduino e sopra un eventuale altro shield.

La RTC shield dispone di una batteria tampone per mantenere l'orario anche senza alimentazione.

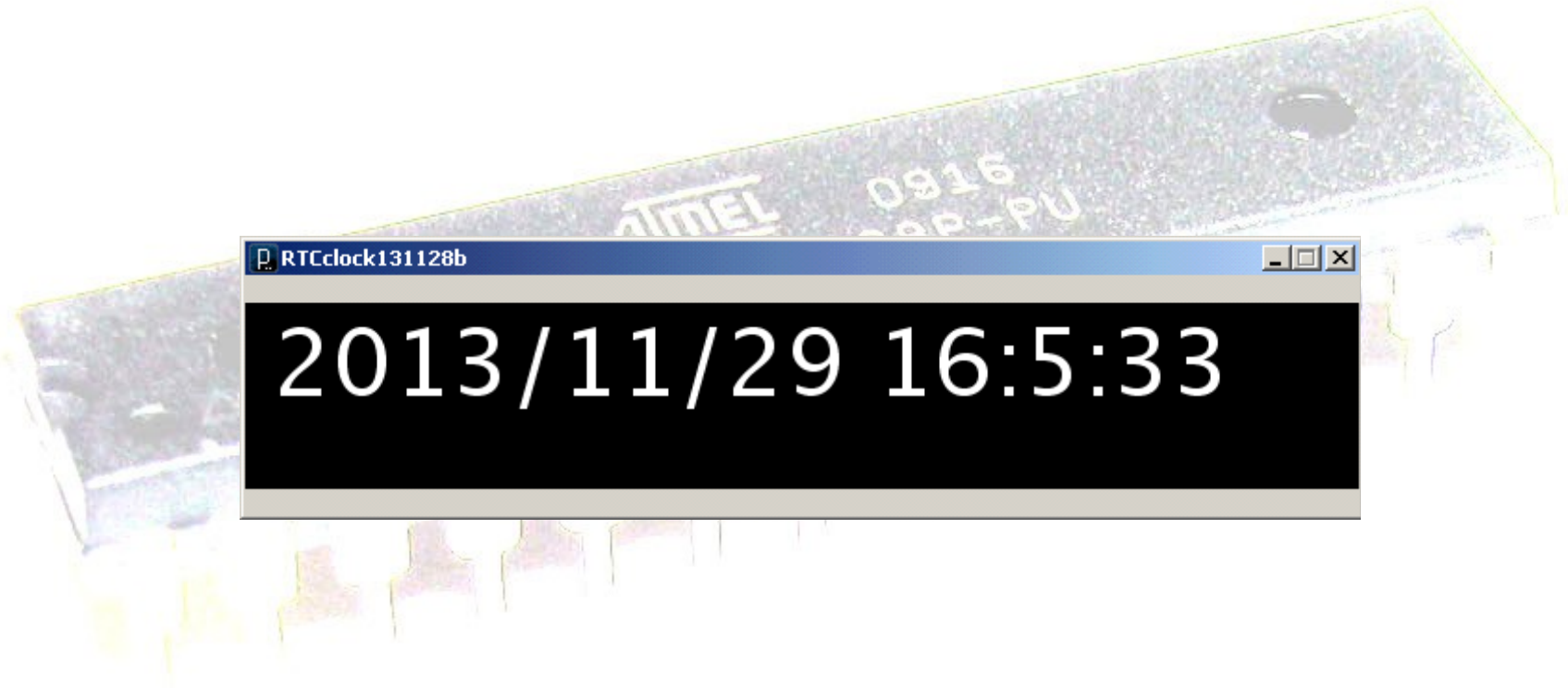
```
#include <Wire.h>   // la scheda utilizza il bus I2C
#include "RTClib.h"  // libreria da aggiungere

RTC_DS1307 RTC;

void setup () {
    Serial.begin(9600);
    Wire.begin();
    RTC.begin();

  if (!RTC.isrunning()) {
    Serial.println("RTC is not  running!");
    // la linea seguente setta   RTC alla data e all'ora della compilazione
    RTC.adjust(DateTime(__DATE__, __TIME__));
  }

RTC.sqw(1); // lampeggio del led ogni secondo
}
```

Crea oggetto di tipo DataTime(variabili di sistema)

```
void loop () {
    DateTime now = RTC.now();

    Serial.print(now.year(), DEC);
    Serial.print('/');
    Serial.print(now.month(), DEC);
    Serial.print('/');
    Serial.print(now.day(), DEC);
    Serial.print(' ');
    Serial.print(now.hour(), DEC);
    Serial.print(':');
    Serial.print(now.minute(), DEC);
    Serial.print(':');
    Serial.print(now.second(), DEC);
    Serial.println();
    delay(1000);
}
```



```
import processing.serial.*;

Serial myPort;  // The serial port

void setup() {
  size(600,100);
  println(Serial.list()); // List all the available serial ports
  background(0);
  myPort = new Serial(this, Serial.list()[2], 9600);
}

void draw() {
  delay(1000);

   background(0);  // nero
  while (myPort.available() > 0) {
    String inBuffer = myPort.readString();
    if (inBuffer != null) {
      fill(255);
      textSize(50);
      text(inBuffer, 15, 50);
    }
  }
}
```


RTCclock131128b
2013/11/29 16:5:33