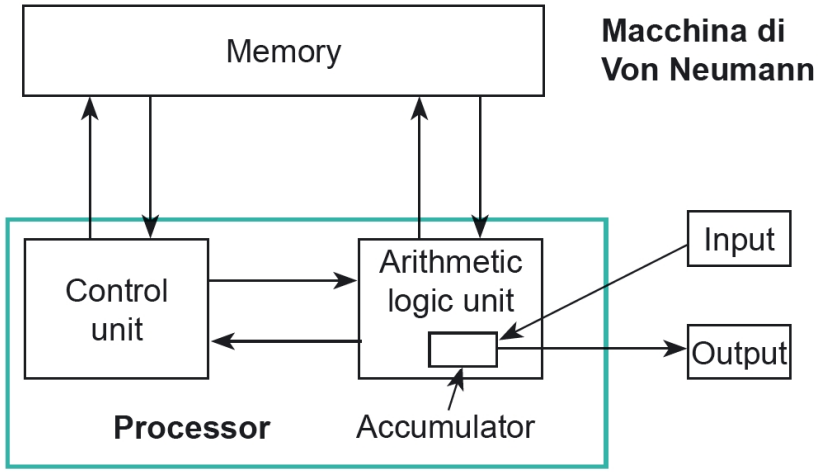
MICROCONTROLLORE

I **Microcontrollori** (MCU: MicroController Unit) sono dispositivi integrati su un singolo chip, che interagiscono direttamente col mondo esterno, grazie a un ***programma residente***.

Normalmente i microcontrollori hanno una struttura interna diversa da quella del personal computer:

Personale computer:
Architettura di Von Neumann

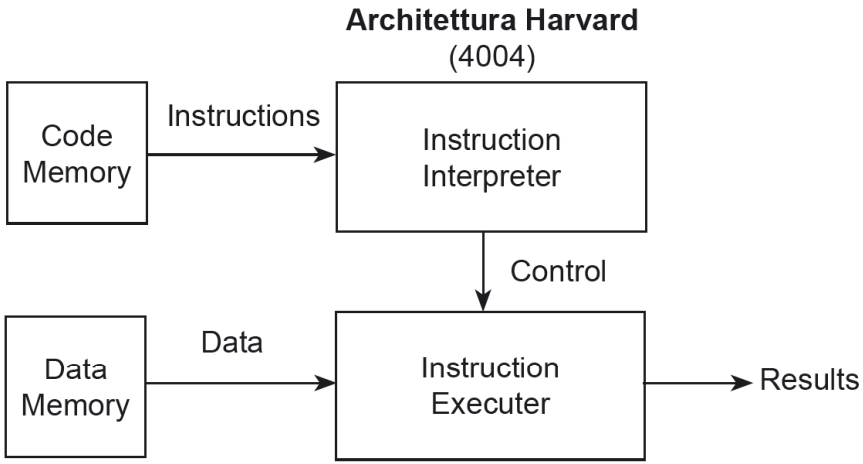


Caratteristiche:

- dati e istruzioni condividono la stessa memoria

NB: maggiore flessibilità nella scrittura del programma.

Microcontrollore:
Architettura Harvard



Caratteristiche:

- dati e istruzioni risiedono in memorie separate

NB: maggiori vincoli nella scrittura del programma, ma a vantaggio della affidabilità.

SCHEDA ARDUINO

La **scheda Arduino** è realizzata con microcontrollori **ATmel,** secondo *l'architettura Harvard.*

Il progetto ARDUINO nasce a Ivrea, nell'*Interaction Design Istitute* (fondato da Olivetti e Telecom Italia), nel 2005, con lo scopo di rendere disponibile per gli studenti uno strumento di controllo economico, rapido e utile ai loro progetti.

Arduino	Processore	Flash	EEPROM	SRAM	I/O digitali	... con PWM	ingressi analogici	Chip USB	Dimensioni
		KB	KB	KB	n. pin	n. pin	n. pin	tipo	mm
Uno	ATmega328P	32	1	2	14	6	6	ATmega8U2	68,6 × 53,3
Diecimila	ATmega168	16	0.5	1	14	6	6	FTDI	68,6 × 53,3
Due (annunciata)	ATMEL SAM3U	256	0	50	54	4	16	-	-
Duemilanove	ATmega168/328P	16/32	0.5/1	1/2	14	6	6	FTDI	68,6 × 53,3
Fio	ATmega328P	32	1	2	14	6	8	Nessuno	40,6 × 27,9
Leonardo	Atmega32u4	32	1	2	14	6	12	Atmega32u4	68,6 × 53,3
LilyPad	ATmega168V or ATmega328V	16	0.5	1	14	6	6	Nessuno	Ø 50
Mega	ATmega1280	128	4	8	54	14	16	FTDI	101,6 × 53,3
Mega2560	ATmega2560	256	4	8	54	14	16	ATmega8U2	101,6 × 53,3
Nano	ATmega168 or ATmega328	16/32	0.5/1	1/2	14	6	8	FTDI	43 × 18

Ruolo delle diverse memorie:

FLASH memory (memoria non volatile): dove Arduino salva il **programma** (Sketch)

In caso di superamento della memoria FLASH, l'IDE genera un messaggio di errore e il programma non viene eseguito.

EEPROM (memoria non volatile): memoria in cui i programmatori possono archiviare informazioni a lungo termine

RAM (SRAM, memoria volatile): memoria dove il programma (sketch) in esecuzione memorizza e manipola le variabili (NB: meno se ne introducono e più memoria si risparmia)

void	1 b	unsigned int	2 B	string – char array	n B
boolean	1 b	word	2 B	String – object	n B
char	1 B	long	4 B	array	n B
unsigned char	1 B	unisgned long	4 B		
byte	1 B	float	4 B		
int	2 B	double	4 B		

In caso di superamento della RAM (anche a causa delle variabili dichiarate nelle librerie utilizzate), cioè di ***overflow,*** non viene fornito alcun messaggio di errore da parte dell'IDE, nonostante la comparsa di seri problemi di funzionamento del programma:

- interruzione dell'esecuzione, oppure
- sovrascrittura delle variabili con conseguenti errori di esecuzione.

Alimentazione (Jack):

Di lavoro	**5 V**
Raccomandata	**7 – 12 V**
Limite	**6 – 20 V**

NB: oltre 12 V il regolatore di tensione tende a surriscaldarsi.

Porta USB:

- **scambio dati con PC**
- **alimentazione della scheda**

NB: in caso di assorbimento di corrente > 500 mA interviene una protezione termica a tutela del PC.

Microcontrollore ATmega 2560

RESET: quando LOW resetta il microcontrollore

Pin AREF
Per modificare il range degli ingressi analogici

Pin uscita
- **3.3 V**
- **max 50 mA**

Pin 13: è connesso a un Led integrato: quando
- **HIGH il Led è ON**
- **LOW il Led è OFF**

Pin uscita
- **5.0 V**
- **corrente max erogabile: sembra < 500 mA (non specificato dal produttore della scheda)**

14 Pin (0 – 13) uscite PWM 8 bit (valori 0 – 255) Max 5 V.
(uscite Analogiche)

VIN: Alimentazione
(Alternativa a Jack e USB)

54 Pin I/O digitali (max 40 mA)
NB: resistenza di pull-up interna 20-50 kΩ (disconnessa di default)
NB: max 5 V

Se pin configurato come IN e si invia *digitalWrite(…, HIGH)* si connette la R di pull-up : il pin presenta valore HIGH finché un circuito esterno lo imposta LOW.

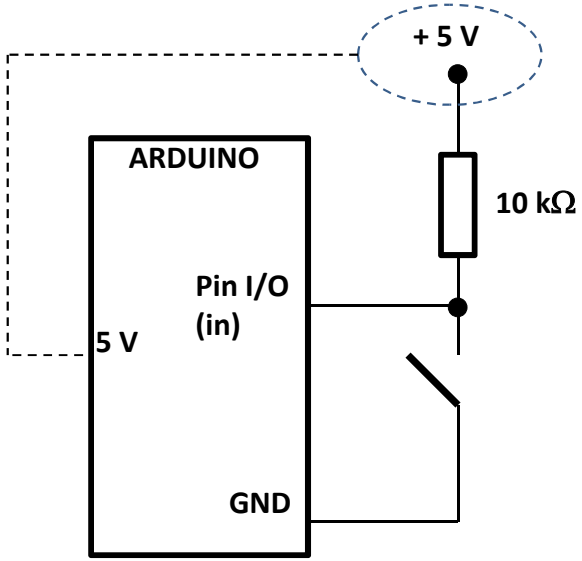
16 Linee di ingresso (A0 – A15) analogiche: conversione a 10 bit (valori 0-1023).
NB: di default si misurano tensioni tra massa e 5 V. Tuttavia è possibile modificare il limite dei 5 V fornendo un diverso riferimento suk pin AREF.

CABLAGGIO

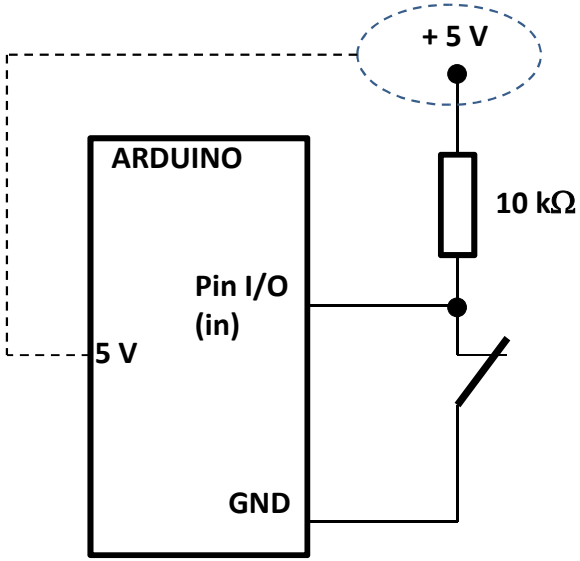
INGRESSI digitali:

- Pulsanti
- Interruttori
- Finecorsa
- Encoder e sensori ON/OFF vari
- uscite da precedenti circuiti digitali

Quando configurati come ingressi, i pin sono posti in uno stato di alta impedenza (100 MΩ).



Normalmente Aperto
- **Premuto: si legge LOW (< 2 V)**
- **A Riposo: si legge HIGH (> 3 V)**

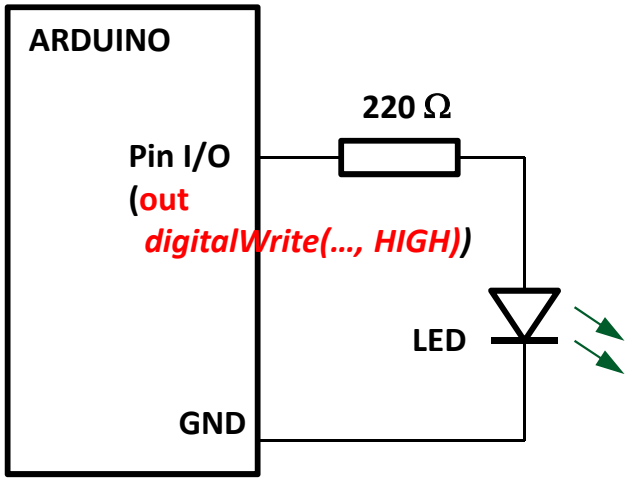


Normalmente Chiuso
- **Premuto: si legge HIGH (> 3 V)**
- **A Riposo: si legge HIGH (< 2 V)**

NB: Anche in caso di alimentazione esterna: V ≤ 5 V

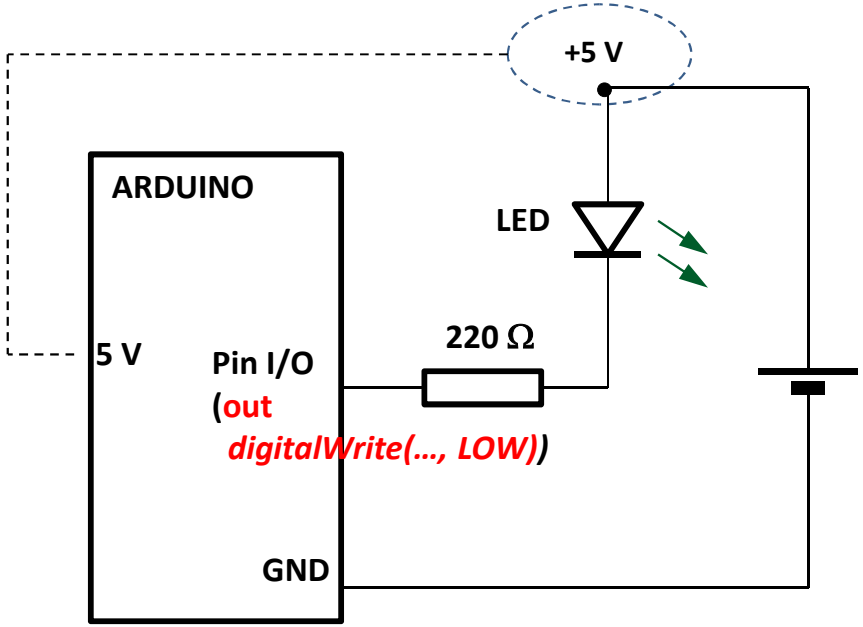
USCITE digitali:

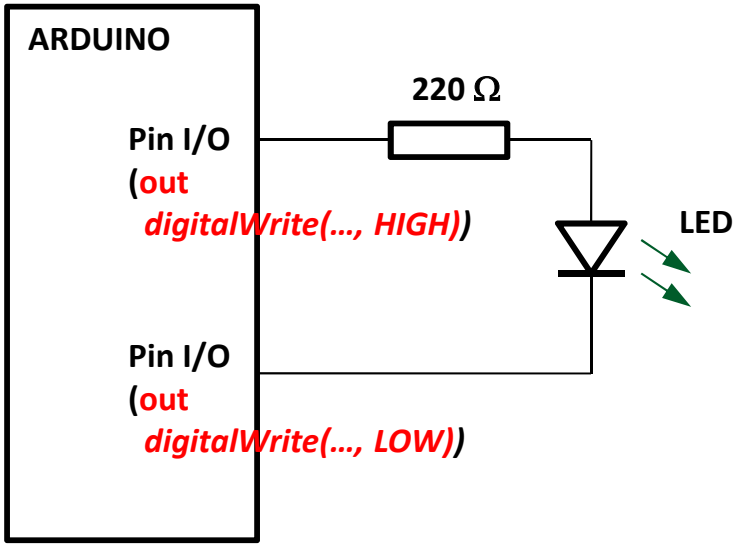
- LED
- Relé
- Verso successivi circuiti digitali



Quando configurati come uscite, i pin sono posti in uno stato di bassa impedenza, con erogazione o assorbimento di corrente ≤ 40 mA.

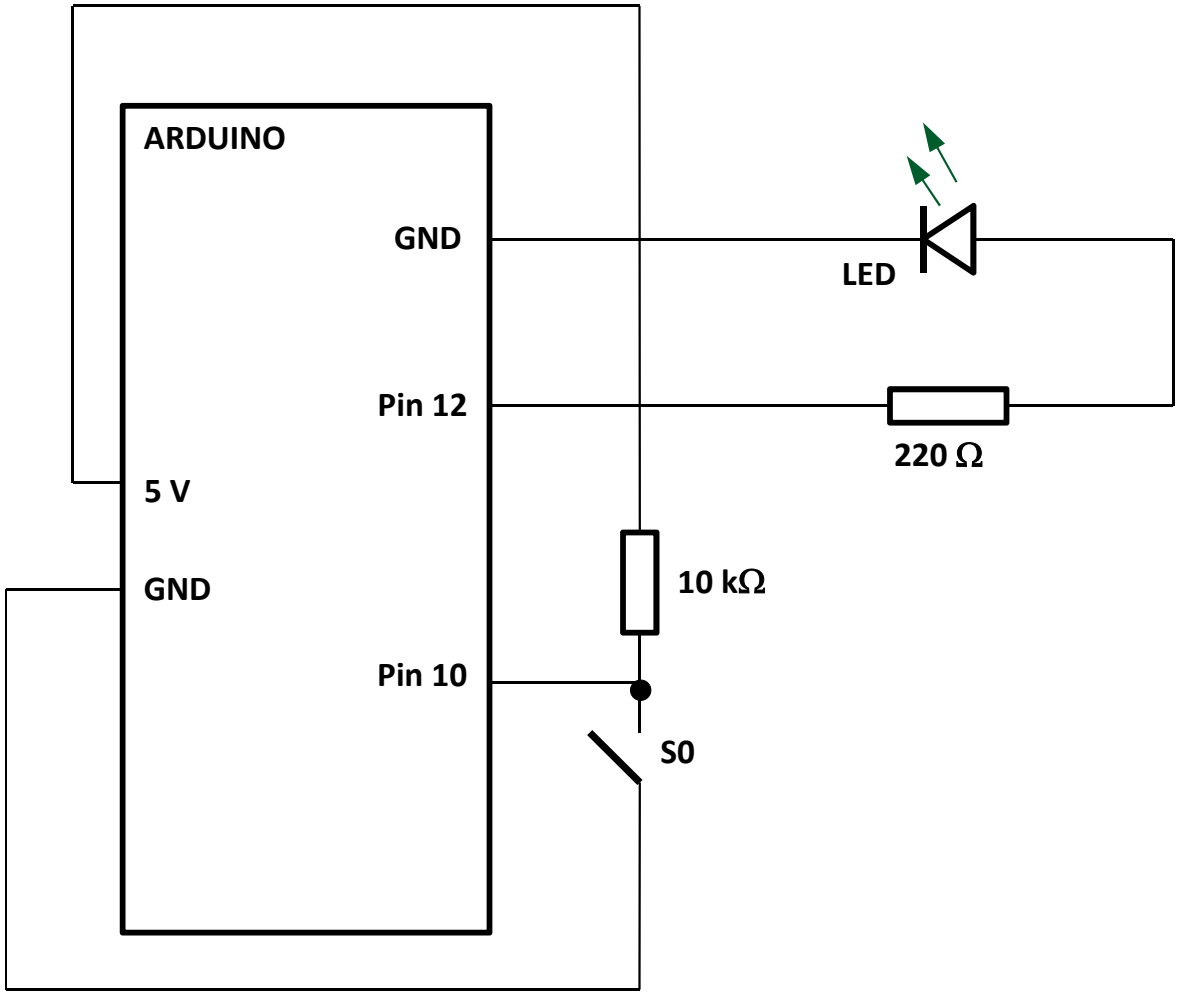
NB: i pin possono essere danneggiati se cortocircuitati a ***massa*** o ***+5 V.***





ESEMPIO. Controllo dell'accensione del LED mediante l'interruttore S0

- S0 aperto LED OFF
- S0 chiuso LED ON



SOFTWARE

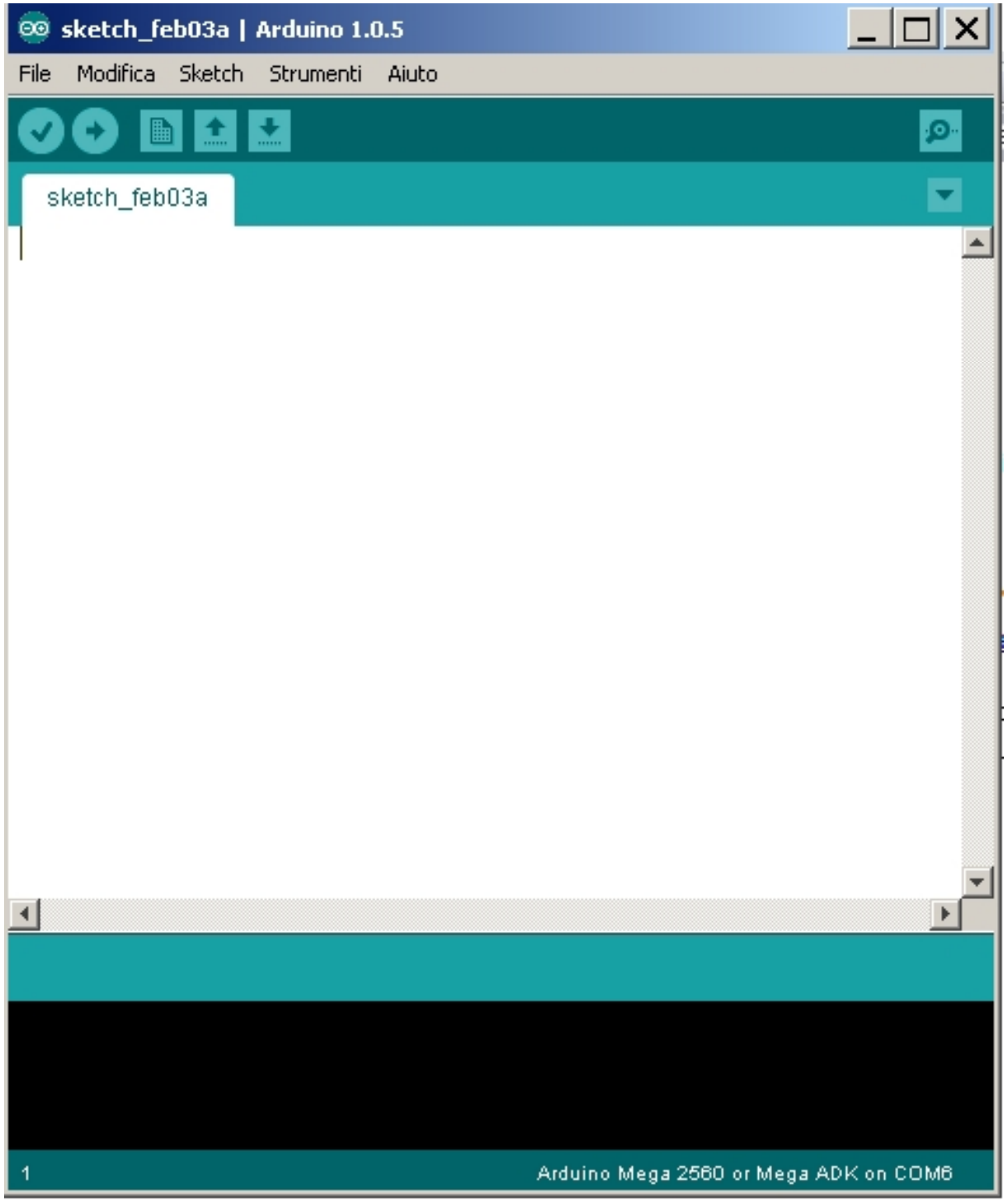
La scheda ARDUINO si programma con un linguaggio che ha la sintassi del C.

I produttori della scheda pongono a disposizione un ambiente di sviluppo (IDE) liberamente scaricabile dal sito:

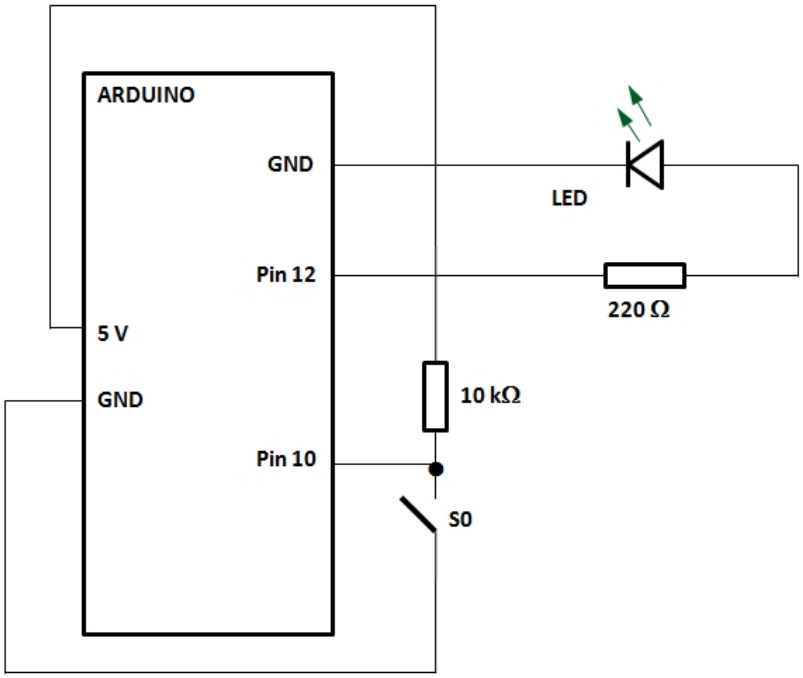
http://arduino.cc/en/Main/Software

Il programma, una volta scritto, è facilmente caricabile nel microcontrollore. Ciò grazie al ***bootlaoder***, un software presente sulla scheda che controlla la presenza di codice sulla porta USB e lo trascrive nel microcontrollore.

I programmi scritti con l'IDE di Arduino sono chiamati ***sketch.***



Sketch di controllo:

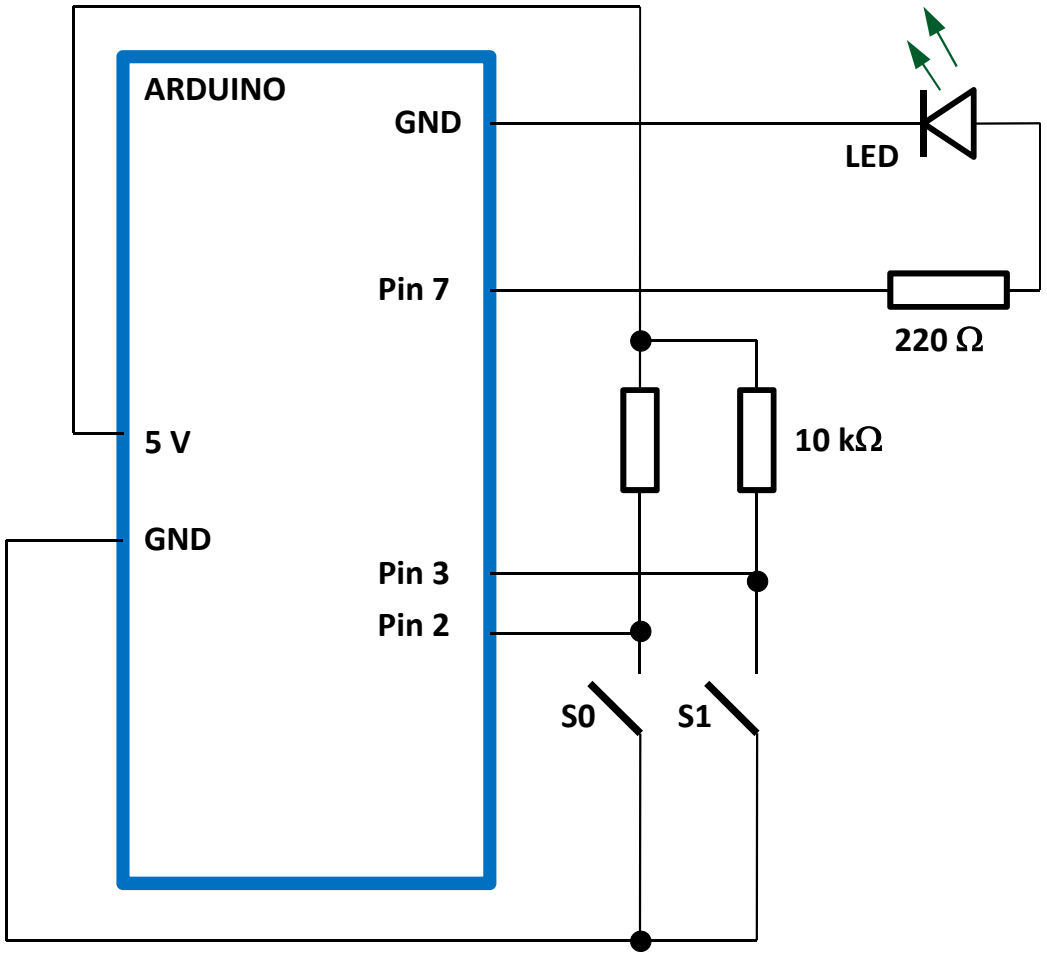



```
// Dichiarazione delle variabili
int pin10 = 10, pin12 = 12;
int in;

// Impostazione dei pin
void setup()
   {pinMode(pin10, INPUT);
    pinMode(pin12, OUTPUT);
    }

// Codice eseguito ciclicamente
void loop()
   {in = digitalRead(pin10);     // lettura stato pin 10
    if (in == HIGH)              // in alternativa  if(in)
        digitalWrite(pin12, HIGH);  // pin 12 High:  LED ON
      else
        digitalWrite(pin12, LOW);   // pin 12 Low:   LED OFF
     }
```

Alternativa:
(con minore impegno di RAM)

```
// Impostazione dei pin
void setup()
   {pinMode(10, INPUT);
    pinMode(12, OUTPUT);
    }

// Codice eseguito ciclicamente
void loop()
   {if (digitalRead(10) == HIGH)
        digitalWrite(12, HIGH);  // pin 12 High:  LED ON
      else
        digitalWrite(12, LOW);   // pin 12 Low:   LED OFF
     }
```

ESEMPIO. Controllo dell'accensione del LED mediante gli interruttori S0 e S1



S0	S1	LED
A	A	OFF
A	C	ON
C	A	ON
C	C	ON

Software di controllo:

```
// Dichiarazione delle variabili
boolean s0, s1;

// Inpostazioni iniziali
void setup()
  {pinMode(2, INPUT);
   pinMode(3, INPUT);
   pinMode(7, OUTPUT);
   }

// Esecuzione ciclica
void loop()
  {s0 = digitalRead(2);
   s1 = digitalRead(3);
   if(s0==HIGH  && s1==HIGH)
            digitalWrite(7, LOW);
   if(s0==HIGH  && s1==LOW)
            digitalWrite(7, HIGH);
   if(s0==LOW  && s1==HIGH)
            digitalWrite(7, HIGH);
   if(s0==LOW  && s1==LOW)
            digitalWrite(7, HIGH);
   }
```

ESEMPIO. Accensione progressiva di un LED (variazione intensità luminosa)

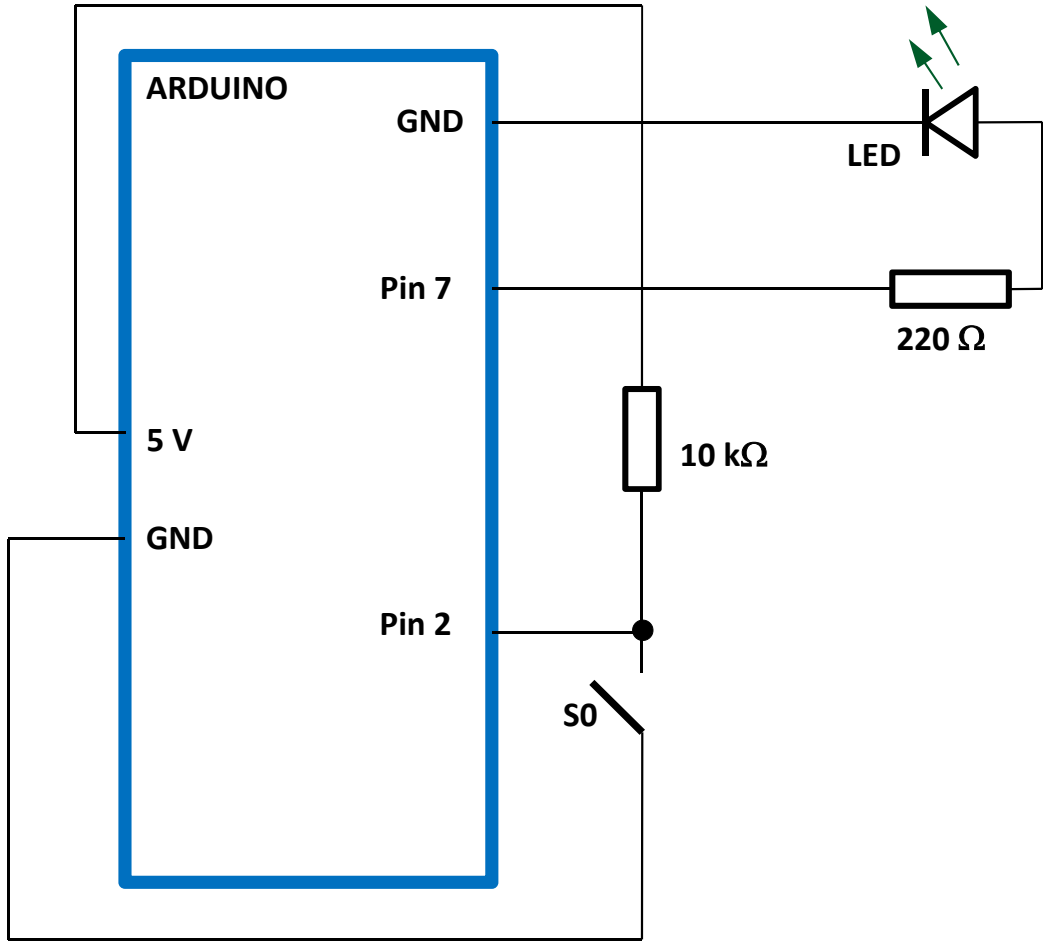
Il LED va controllato con uscita analogica.

I pin 0 - 13 possono lavorare anche come uscite **PWM** (8 bit, cioè 256 livelli di tensione selezionabili: **0 - 255**).

Range di tensione fornita: **0 - 5 V.**

Risoluzione: 5/255 = **19.6 mV.**

Non ha senso inviare livelli superiori a 255.
Esempio: 258 corrisponde a 258 – 255 = 3.



Software di controllo:

```
// Dichiarazioni
boolean s0;
int x = 0;

// Impostazioni iniziali
void setup()
 { pinMode(2, INPUT);
   pinMode(7, OUTPUT);
  }

// Esecuzione ciclica
void loop()
  { s0 = digitalRead(2);
    if(s0 == LOW)
       {analogWrite(7, x); // uscita PWM7
        delay(100);       // ritardo di 100 ms
        x++;
        if(x > 255)
          x = 255;    // 256 corrisponde a 0
                      // 257    "        a 1
        }
      else
        analogWrite(7,0);
   }
```

Lo **sketch** legge continuamente S0:

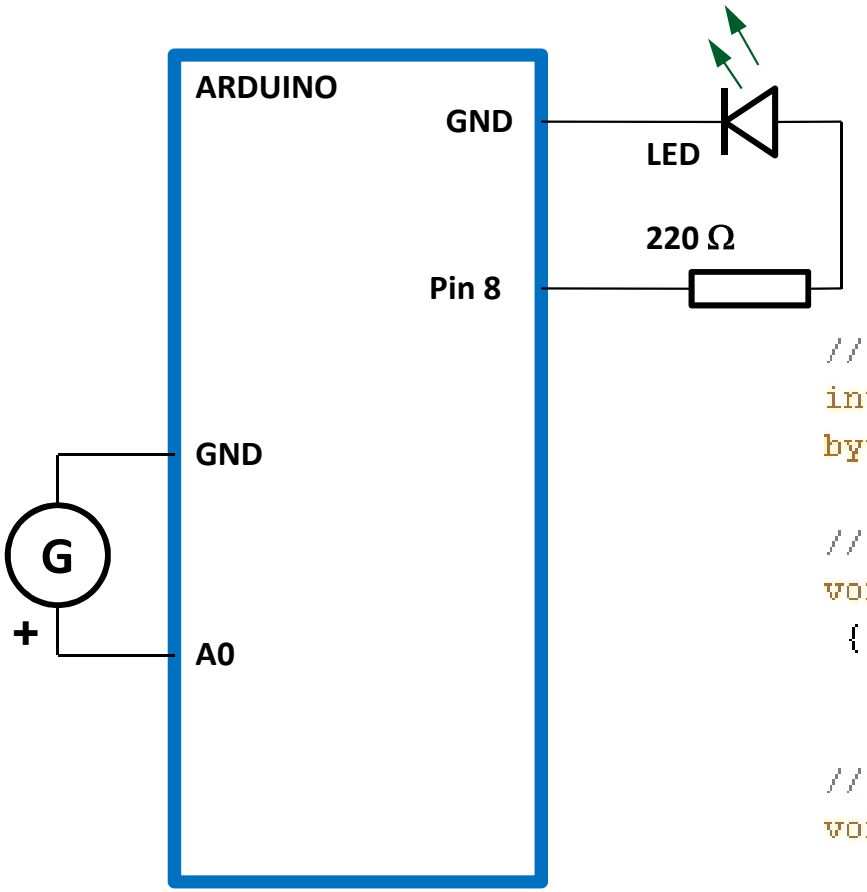
- finché S0 è LOW sul pin 7 viene inviata una tensione che aumenta progressivamente;
- quando S0 è HIGH, al pin 7 è inviata una tensione pari a 0;
- al ritorno di S0 LOW al pin 7 si ripresenta il valore di tensione e riprende il suo aumento progressivo, fino a 255 (5 V).

ESEMPIO. Lettura di un ingresso analogico per controllo luminosità LED

Range di ingresso: **0 – 5 V**.

Il convertitore A/D è a 10 bit: 2^{10} = 1024 livelli di tensione in ingresso (**0 – 1023**).

Risoluzione: 5/1023 = **4.9 mV**

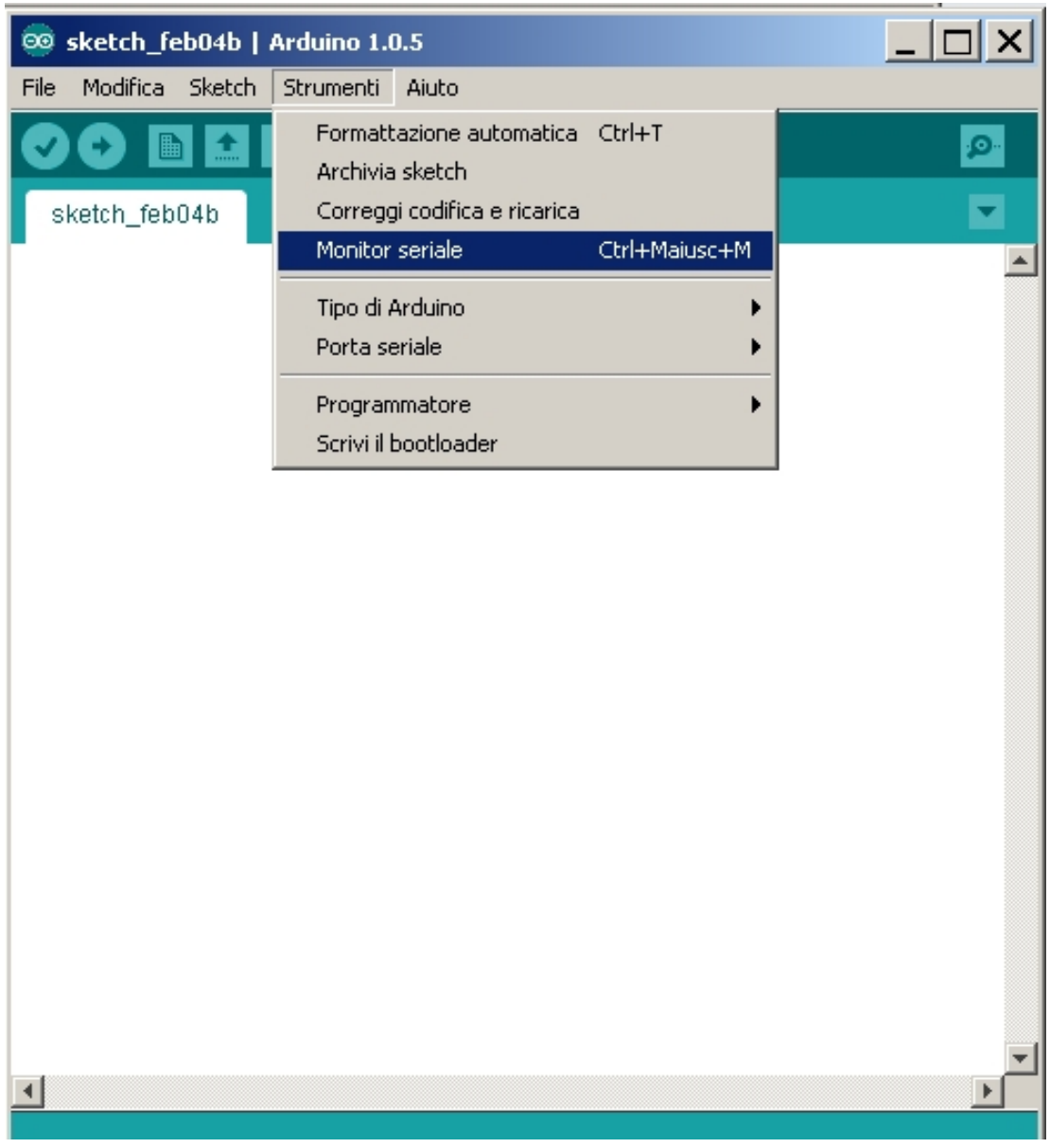


Software di controllo:

```
// Dichiarazioni
int x = 0;   // variabile di 16 bit
byte y = 0;   // variabile di 8 bit

// Impostazioni iniziali
void setup()
 { pinMode(8, OUTPUT);
  }

// Esecuzione ciclica
void loop()
  { x = analogRead(A0);     // lettura canale analogico A0
    y = map(x, 0, 1023, 0, 255);      // adattamento del
                                 // range 0 - 1023 al range pwm 0 - 255
    analogWrite(8, y);     // invio valore pwm al controllo LED
    }
```

COMUNICAZIONI SERIALI (USB)

L'IDE di Arduino permette una comunicazione seriale USB per visualizzare su PC i valori delle variabili di interesse.



L'apertura del **Monitor seriale** attiva il collegamento USB.

I pin 0 e 1 non sono utilizzabili come I/O quando il **Monitor seriale** è attivo.

Sul **Monitor seriale** occorre impostare la stessa velocità impostata nello sketch.

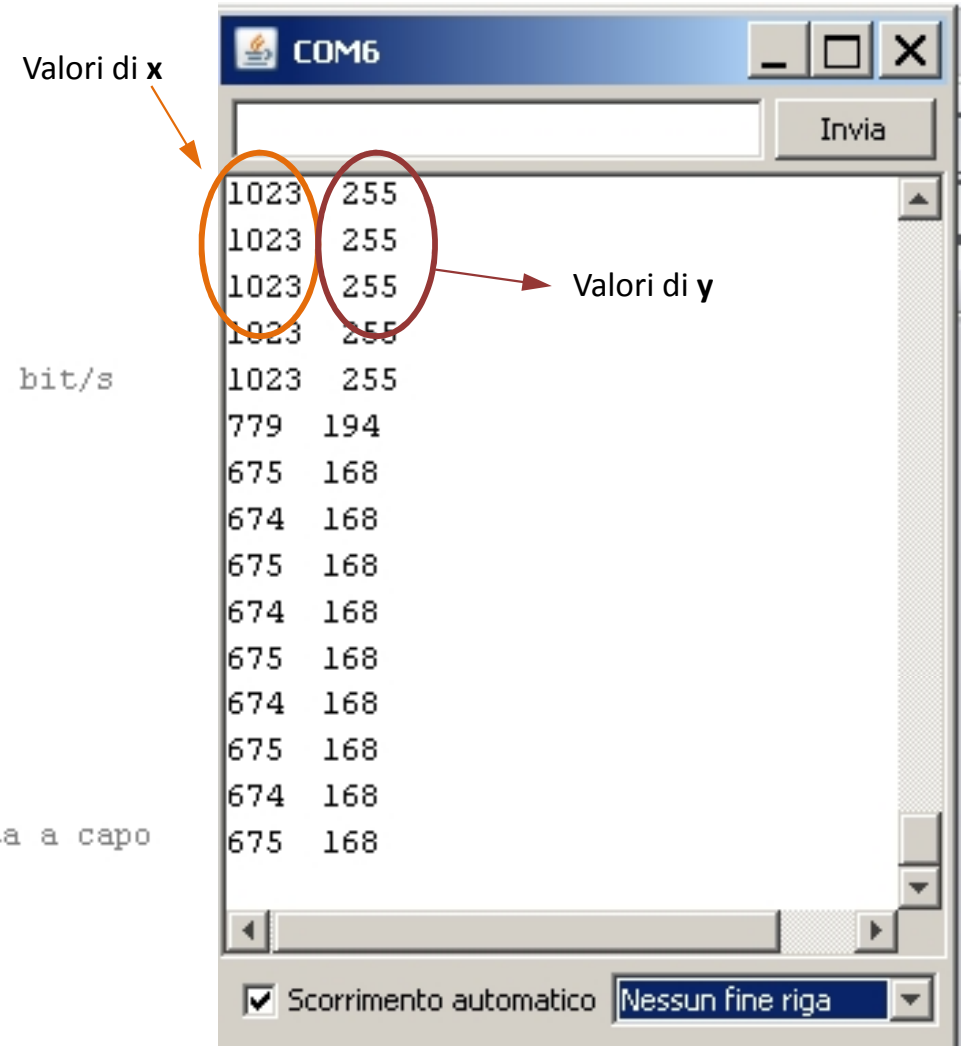
Procedura:

- Caricare lo sketch in Arduino
- Aprire il Monitor seriale

ESEMPIO. Circuito precedente, con visualizzazione dei valori delle variabili x e y

Software di controllo:

```
// Dichiarazioni
int x = 0;
byte y = 0;

// Impostazioni iniziali
void setup()
 { pinMode(8, OUTPUT);
   Serial.begin(9600);  // impostazione velocità USB: 9600 bit/s
  }

// Esecuzione ciclica
void loop()
  { x = analogRead(A0);
    y = map(x, 0, 1023, 0, 255);
    analogWrite(8, y);
    Serial.print(x);       // scrittura valore di x
    Serial.print("  ");    // scrittura di spazio bianco
    Serial.println(y);     // scrittura valore di y e andata a capo
    delay(1000);
    }
```

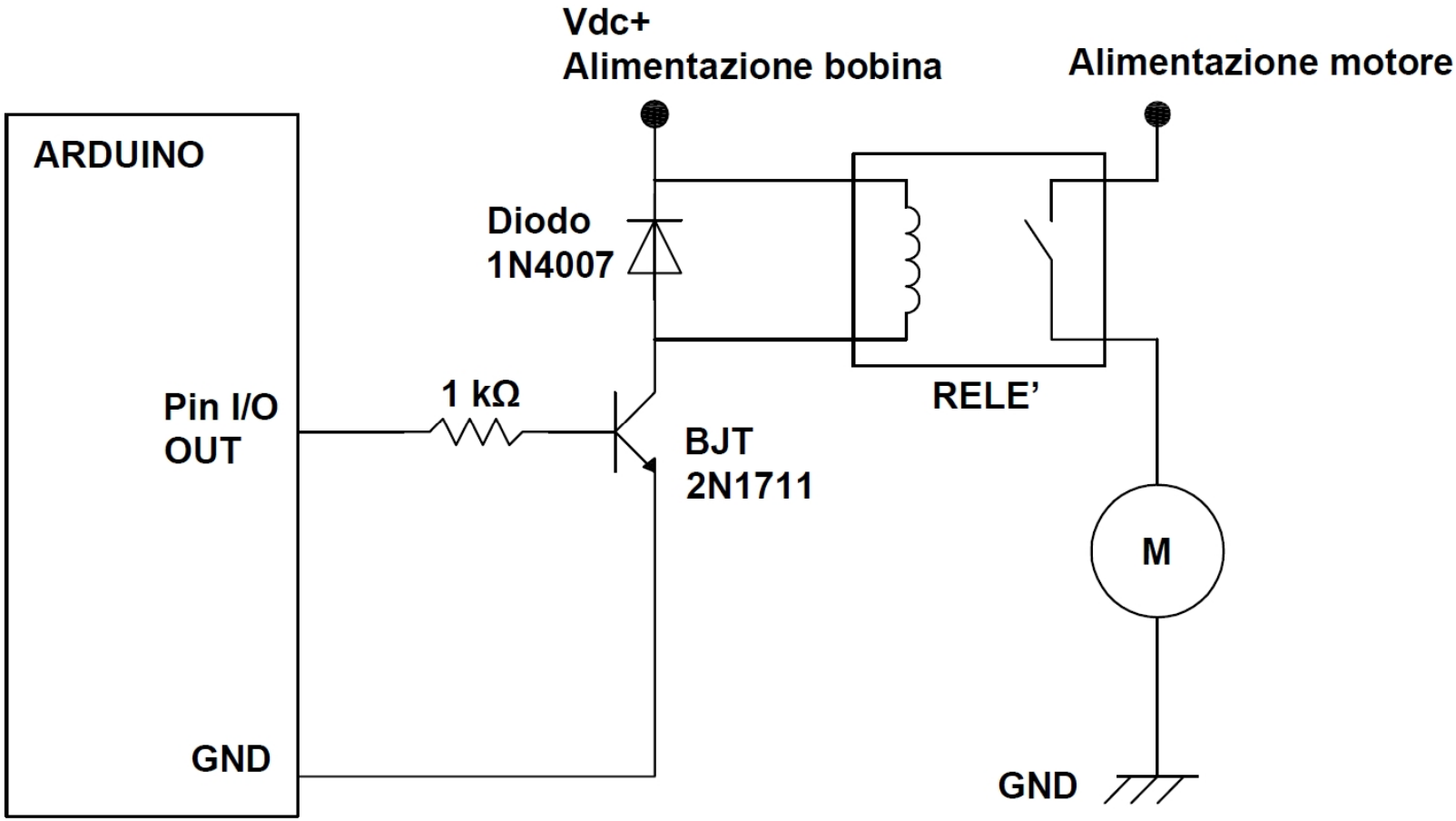


ESEMPIO. Controllo marcia/arresto di un motore

Quando la bobina:

- richiede una alimentazione > 5 V, oppure
- assorbe una corrente > 40 mA

È necessario interporre tra Arduino e bobina uno stadio di interfaccia.



Descrizione:

Transistor **BJT** npn (**2N1711**)

BJT ON bobina eccitata, motore in marcia

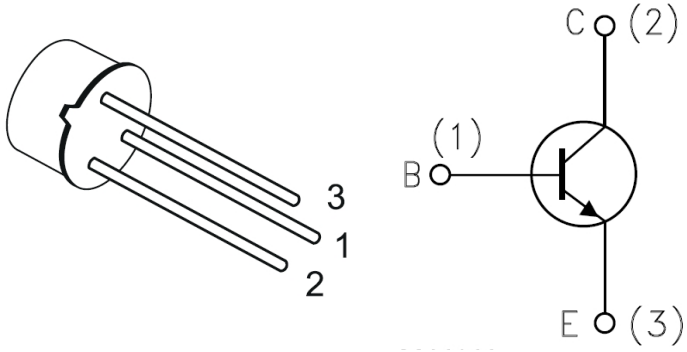
Per accensione BJT necessaria : $(V_{BE})_{SAT} \sim 0.9 - 1.3$ V

Tensione di controllo fornita da Arduino: 5 V, quindi $I_B \sim (5 - 0.9) / 1k = 4.1$ mA < **40 mA**

(NB: $I_{BOBINA} < (I_{BJT})_{MAX}$ = **500 mAdc** per 2N1711)

BJT OFF bobina diseccitata, motore fermo

(NB : $V_{BOBINA} < (V_{BJT\_CE\_OFF})_{MAX}$ = **80 Vdc** 2N1711)



Diodo (**1N4007**)

Ha funzione protettiva (per BJT e bobina) contro la sovratensione provocata dalla bobina al momento dello spegnimento del BJT ($e = - d\Phi/dt$)

Bobina

Rappresenta l'interfaccia di potenza. Occorre prestare attenzione alla corrente assorbita dal motore, che deve essere sopportabile dal contatto di potenza del relé.

Motore

Può essere sia DC che AC.