Un attuatore lineare per aprire uno sportello

Il pistoncino elettrico DFRobot da 6 V con Arduino UNO e il driver DRV8833: due pulsanti, la corrente letta dal driver per fermarsi in fondo, le pause di riposo.

A cura di **Ada Intelisano** · redazione tecnica di informaticait.it

Guida online, sempre aggiornata

IN BREVE

- L'attuatore è un motorino che spinge uno stelo: scambiando i due fili lo stelo esce o rientra. DFRobot dichiara 6 V e 128 N di spinta; sull'etichetta c'è anche 7 mm/s.
- DFRobot non dice quanta corrente assorbe né se dentro ci sono finecorsa: per questo usiamo il DRV8833 di Adafruit, che limita la corrente a circa 1 A per ponte.
- Dal pin AS del driver esce una tensione di 0,2 V per ampere: Arduino la legge e ferma il motore quando lo stelo arriva in fondo o trova un ostacolo.
- Sull'etichetta c'è un 10 %: lo leggiamo come ciclo di lavoro, e il programma fa riposare l'attuatore nove volte il tempo in cui si è mosso.

Tempo: un pomeriggio | Livello: un passo avanti | Prima: guide n. 35 e 49

Un attuatore lineare trasforma il giro di un motorino in una spinta in linea retta: lo stelo esce, si ferma, rientra. Con uno di questi apri lo sportello di una scatola, una botola, una presa d'aria. Qui usiamo gli attuatori DFRobot da 6 V, che esistono con quattro corse diverse, e li comandiamo da Arduino UNO con due pulsanti, APRI e CHIUDI. La parte più interessante è quella che non si vede: come fermarsi in fondo senza sapere quanta corrente assorbe il motore, perché DFRobot non lo scrive.

1 Il materiale

- un **attuatore lineare DFRobot** da 6 V: FIT0803 (corsa 10 mm), FIT0804 (30 mm), FIT0805 (50 mm) o FIT0806 (100 mm). Negli esempi c'è il FIT0805;
- **Arduino UNO R3** con il cavo USB;
- il driver **Adafruit DRV8833** (3297), con il pettine da saldare;
- un **portapile da 4 pile AA** con interruttore (Adafruit 830) e quattro pile;
- due pulsanti da 6 mm, una breadboard, cavetti maschio-maschio.

Parola nuova: attuatore lineare
Un motorino in continua che, attraverso una vite, fa scorrere uno stelo. Il verso dipende da come arriva la corrente ai due fili: scambiandoli, lo stelo che usciva rientra.

2 Com'è fatto e quanto si allunga

I quattro modelli hanno lo stesso corpo; cambia la corsa, cioè quanto esce lo stelo. Per montarlo conta l'interasse dei due fori da 4 mm, quello dell'orecchia sul corpo e quello

dell'occhiello dello stelo. Dal disegno di DFRobot: da chiuso è 54,5 mm più la corsa, da aperto c'è una corsa in più. Il corpo è alto 20 mm, il tubo 16. Il cavo è rosso e nero e finisce in un connettore femmina a due poli.

L'attuatore chiuso e aperto, con le misure di DFRobot (qui la corsa da 50 mm)



informaticait.it

Modello	Corsa	Interasse dei fori da chiuso	Da aperto	Una corsa a 7 mm/s (calcolo)
FIT0803	10 mm	64,5 mm	74,5 mm	1,4 s
FIT0804	30 mm	84,5 mm	114,5 mm	4,3 s
FIT0805	50 mm	104,5 mm	154,5 mm	7,1 s
FIT0806	100 mm	154,5 mm	254,5 mm	14,3 s

DFRobot dichiara 6 V e una spinta di 128 N, più o meno il peso di 13 kg (calcolo nostro). La velocità, 7 mm/s, è nella pagina del FIT0803 e sulle etichette di tutti e quattro, senza dire con quale carico: i tempi della tabella sono solo indicativi, e con un peso da spingere lo stelo può andare più piano. L'etichetta riporta anche IP54, un grado di protezione contro polvere e acqua che vale per l'attuatore, non per Arduino e il driver.

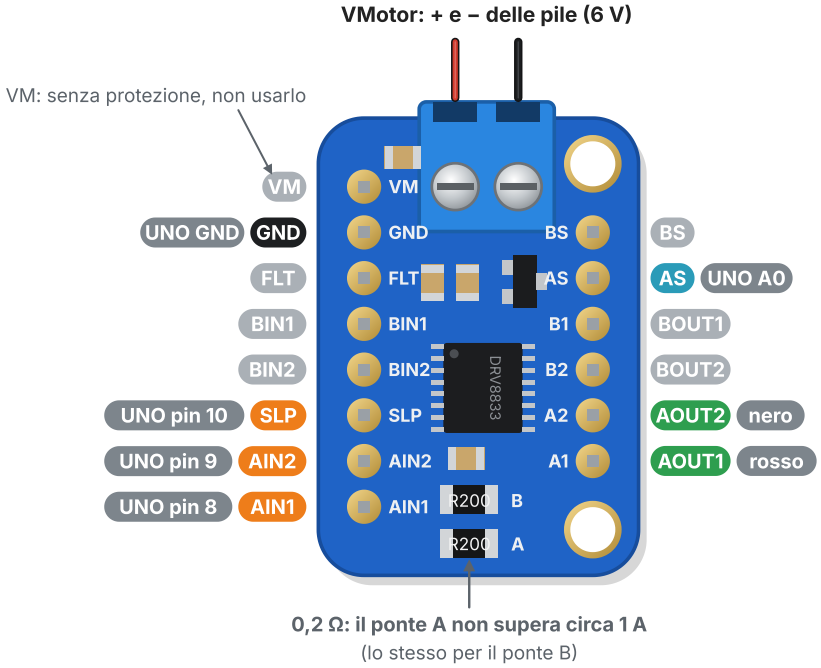
Quello che DFRobot **non** dice conta quanto il resto: quanta corrente assorbe, a vuoto e con lo stelo bloccato; se dentro ci sono finecorsa che spengono il motore in fondo; in che verso va lo stelo con il rosso sul +. C'è poi un «10%», che nella pagina del FIT0803 si chiama «Frequency». Noi lo leggiamo come ciclo di lavoro: l'attuatore si muove al massimo per un decimo del tempo e per il resto riposa. È una nostra interpretazione, DFRobot non la spiega.

3 Perché il DRV8833

Un pin di Arduino non regge un motore (guida n. 49), e qui serve anche invertire il verso: ci vuole un ponte H. Sceglierlo è facile quando si conosce la corrente del motore. Qui non la conosciamo, quindi cerchiamo un driver che si protegga da solo e che protegga l'attuatore.

Driver a catalogo	Con questo attuatore
Adafruit DRV8833 (3297)	Motori da 2,7 a 10,8 V. Limite di corrente già montato, circa 1 A per ponte; protezioni contro sovracorrente, surriscaldamento e tensione bassa; il pin AS per leggere la corrente. **La nostra scelta.**
TB6612FNG (Adafruit 2448, DFRobot DRI0044, SparkFun 14450)	Tensione giusta, 1,2 A medi e 3,2 A di picco. Toshiba dichiara la protezione termica e quella per tensione bassa, non un limite di corrente: va bene quando la corrente del motore è nota.
DRV8871 (Adafruit 3190)	Lavora da 6,5 V in su: a 6 V è sotto il minimo.
L9110H (Adafruit 4489)	Adafruit lo dà per 800 mA: poco margine con una corrente che non conosciamo.

Adafruit DRV8833 vista dall'alto: in colore i pin che usiamo



informaticait.it

Sulla scheda Adafruit due resistenze da 0,2 Ω misurano la corrente dei due ponti. Quando la tensione su una di loro arriva a 0,2 V il chip stacca il ponte per un attimo, fino al ciclo successivo del suo PWM interno (TI): 0,2 V diviso 0,2 Ω fa 1 A, il limite impostato da Adafruit. Se lo stelo resta bloccato in fondo, il motore riceve al massimo circa 1 A invece di tutta quella che chiederebbe. Lo stesso punto esce sul pin **AS**: Arduino lo legge con A0 e sa quanta corrente passa.

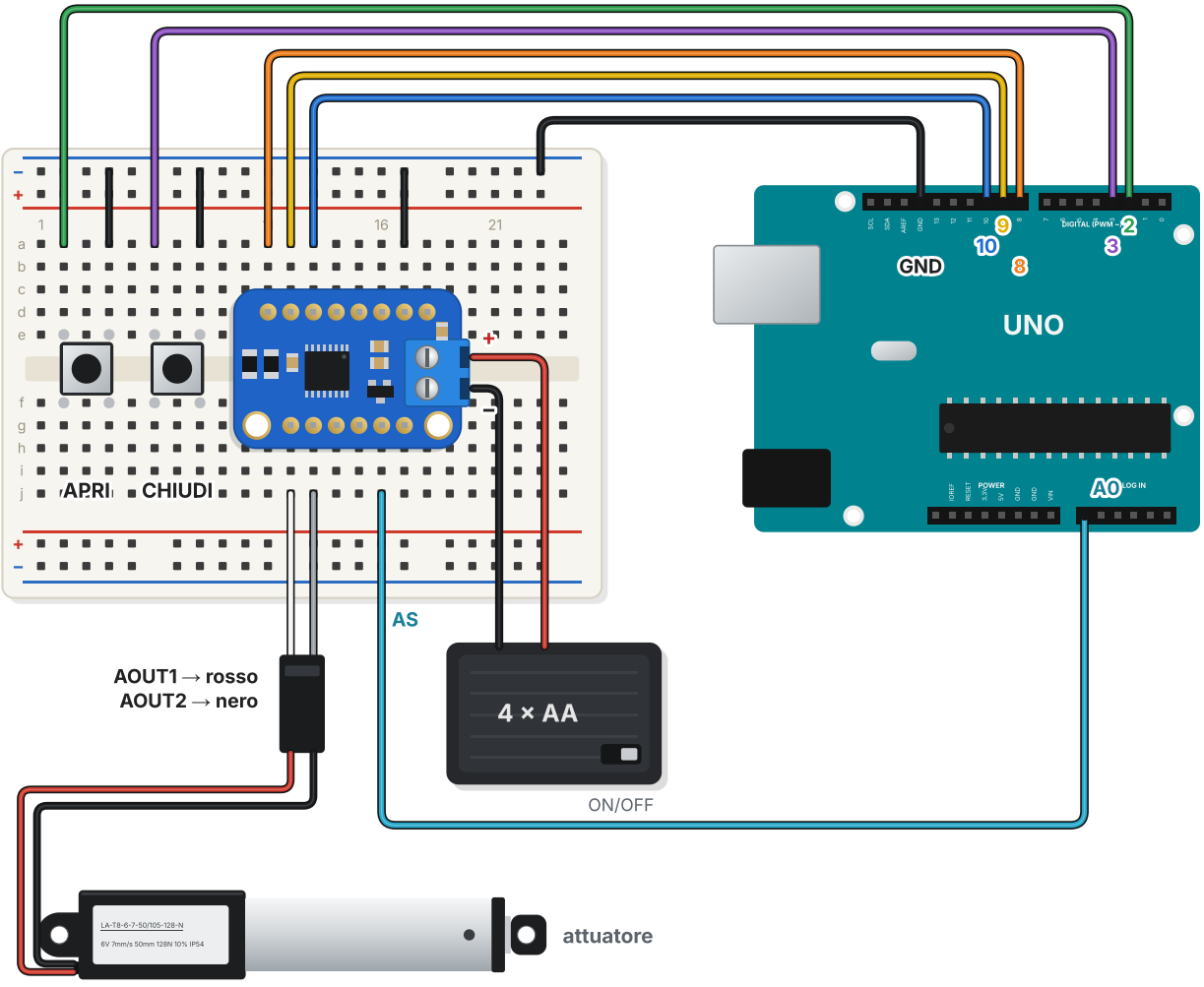
Il pin **SLP** è tenuto basso sulla scheda, e finché resta basso il driver dorme. Lo comandiamo con il pin 10: acceso solo mentre lo stelo si muove, così il chip a riposo consuma pochi microampere e le pile durano di più.

> **Parola nuova: ponte H**
> Quattro interruttori elettronici disposti ad H attorno al motore. Chiudendone due in diagonale la corrente passa in un verso, con gli altri due passa al contrario. Il DRV8833 ne ha due, A e B: noi usiamo il ponte A, comandato da AIN1 e AIN2.

AIN1	AIN2	Il motore
0	0	libero: le uscite si staccano
1	0	gira in un verso (AOUT1 alto, AOUT2 basso)
0	1	gira nell'altro verso
1	1	frenato: tutte e due le uscite basse

4 I collegamenti

I collegamenti: pulsanti, driver, attuatore e pile



informaticait.it

- **Pulsanti** a cavallo del canale: APRI nelle colonne 2 e 4, CHIUDI nelle colonne 6 e 8. Pin 2 ad APRI, pin 3 a CHIUDI; l'altro piedino di ogni pulsante va alla fila −.
- **Driver** a cavallo del canale, con il morsetto verso destra: il pettine nella riga d (AIN1 nella colonna 11, VM nella 18), la fila con AOUT e AS nella riga g.
- Pin **8** → AIN1, pin **9** → AIN2, pin **10** → SLP. Il GND del driver va alla fila −, e la fila − al GND della UNO.
- **AS** (colonna 16, riga j) → **A0**, con il filo azzurro che passa sotto.
- **Attuatore**: due cavetti dal driver al suo connettore, AOUT1 sul rosso e AOUT2 sul nero.

- **Pile** nel morsetto VMotor: rosso al +, nero al −. Tieni spento l'interruttore finché non hai controllato tutto.

> **Attenzione**
> - Le pile vanno nel morsetto VMotor, che ha la protezione contro l'inversione. Il pin VM del pettine non è protetto (Adafruit).
> - L'attuatore non prende mai corrente dal 5V di Arduino: la corrente del motore arriva solo dalle pile.
> - Tutti i GND in comune: driver, fila −, UNO. Senza, A0 legge numeri a caso.

5 La prima prova: fuori, dentro e la corrente

Metti in `CORSA_MM` la corsa del tuo modello. Il programma calcola il tempo di una corsa a 7 mm/s, aggiunge il 30 % (margine nostro), fa uscire lo stelo, aspetta tre secondi e lo fa rientrare. Intanto, ogni 100 ms, scrive sul monitor seriale (115200 baud) il tempo e la corrente.

> **Parola nuova: analogReference(INTERNAL)**
> Di solito `analogRead` misura da 0 a 5 V in 1024 passi, quasi 5 mV per passo. Sul pin AS ci sono pochi decimi di volt, quindi passiamo al riferimento interno della UNO, 1,1 V: ogni passo vale circa 1 mV. Arduino avverte che dopo il cambio le prime letture non sono precise, e il programma ne scarta cinque.

```
// Prova dell'attuatore lineare: lo stelo esce, si ferma, rientra. Ogni 100 ms il
// monitor seriale mostra la corrente letta sul pin AS del driver DRV8833.
#include <Arduino.h>

const byte AIN1 = 8;          // i due ingressi del ponte A
const byte AIN2 = 9;
const byte SLP = 10;          // HIGH = driver acceso, LOW = driver a riposo
const byte SENSE = A0;        // pin AS: 0,2 V per ogni ampere che passa nel motore

const unsigned int CORSA_MM = 50;   // la corsa del tuo attuatore: 10, 30, 50 o 100
// durata di una corsa: corsa diviso 7 mm/s (dato DFRobot), con il 30 % in più di margine
const unsigned long TEMPO_MS = CORSA_MM * 1300UL / 7;

int correnteMA() {
  long somma = 0;
  for (byte i = 0; i < 20; i++) somma += analogRead(SENSE);
  float mV = somma / 20.0 * 1100.0 / 1024.0;   // media in millivolt, scala da 1,1 V
  return (int)(mV * 5);                        // 1 mV su 0,2 ohm = 5 mA
}

void muovi(int verso) {       // 1 = fuori, -1 = dentro, 0 = fermo
  digitalWrite(AIN1, verso > 0 ? HIGH : LOW);
  digitalWrite(AIN2, verso < 0 ? HIGH : LOW);
}

void corsa(int verso) {
```

```
  Serial.println(verso > 0 ? "FUORI" : "DENTRO");
  digitalWrite(SLP, HIGH);
  delay(2);                    // il DRV8833 si sveglia entro 1 ms
  muovi(verso);
  unsigned long inizio = millis();
  while (millis() - inizio < TEMPO_MS) {
    char riga[32];
    snprintf(riga, sizeof riga, "%5lu ms  %4d mA", millis() - inizio, correnteMA());
    Serial.println(riga);
    delay(100);
  }
  muovi(0);
  digitalWrite(SLP, LOW);
}

void setup() {
  pinMode(AIN1, OUTPUT);
  pinMode(AIN2, OUTPUT);
  pinMode(SLP, OUTPUT);
  Serial.begin(115200);
  analogReference(INTERNAL);                      // scala da 0 a 1,1 V, non da 0 a 5 V
  for (byte i = 0; i < 5; i++) analogRead(SENSE);  // le prime letture non valgono
  corsa(1);
  delay(3000);
  corsa(-1);
  Serial.println("Fine. Per ripetere premi RESET, dopo qualche minuto di riposo.");
}

void loop() {
}
```

6 Leggi i numeri

All'avvio vedi un picco breve: è lo spunto del motore. Poi la corrente scende e resta più o meno ferma mentre lo stelo corre. Guarda bene cosa succede quando lo stelo arriva in fondo:

- la corrente **crolla a zero**: dentro l'attuatore c'è un finecorsa che stacca il motore;
- la corrente **sale e resta alta**: il motore spinge contro il fondo e il driver la tiene sotto il suo limite. Meglio fermarsi prima, ed è quello che fa il programma del passo 7.

Annota due numeri: la corrente durante la corsa e quella in fondo. Servono per la soglia. Sono stime, perché il riferimento interno della UNO non è preciso come uno strumento: per un valore vero usa il tester in serie (guida n. 8).

> **✓ Cosa devi vedere**
> Lo stelo esce, si ferma tre secondi e rientra. Nel monitor scorrono righe con il tempo in millisecondi e la corrente in milliampere.

Se non funziona

- Non si muove e la corrente è 0: interruttore del portapile spento, SLP non arriva al pin 10, oppure manca il GND in comune.
- Con «FUORI» lo stelo rientra: scambia i due cavetti nel connettore dell'attuatore.
- Si muove ma la corrente resta 0: il filo azzurro è su BS invece che su AS, o non arriva ad A0.
- Prima di ripetere la prova lascia riposare l'attuatore qualche minuto: è il 10 % del passo 2.

7 Lo sportello con due pulsanti

APRI fa uscire lo stelo, CHIUDI lo fa rientrare. Il motore si ferma per il primo di tre motivi:

- tocchi un pulsante qualsiasi mentre si muove;
- la corrente resta sopra `SOGLIA_MA` per più di 150 ms: lo stelo è in fondo o ha trovato un ostacolo. Nei primi 300 ms non conta, perché allo spunto la corrente è alta comunque;
- passa il tempo massimo, calcolato come nella prova.

Poi l'attuatore riposa per nove volte il tempo in cui si è mosso, e intanto il LED L lampeggia: è il 10 % dell'etichetta letto come ciclo di lavoro (proposta nostra). Dopo una corsa di 7 secondi, circa un minuto di pausa. Il programma ricorda anche dove si è fermato: da aperto ignora APRI, da chiuso ignora CHIUDI. All'accensione non lo sa e accetta tutti e due.

```
// Lo sportello: APRI e CHIUDI con due pulsanti. Il motore si ferma per tre motivi:
// un tocco su un pulsante, la corrente troppo alta (stelo in fondo o contro un
// ostacolo) o il tempo massimo. Dopo ogni corsa l'attuatore riposa.
#include <Arduino.h>

const byte P_APRI = 2;        // pulsanti verso GND, con il pull-up interno
const byte P_CHIUDI = 3;
const byte AIN1 = 8;
const byte AIN2 = 9;
const byte SLP = 10;
const byte SENSE = A0;        // pin AS del driver: 0,2 V per ogni ampere

const unsigned int CORSA_MM = 50;                       // la corsa del tuo attuatore
const unsigned long TEMPO_MAX = CORSA_MM * 1300UL / 7;  // corsa / 7 mm/s, +30 %
const int SOGLIA_MA = 800;           // oltre questa corrente lo stelo è bloccato
const unsigned long SPUNTO_MS = 300; // alla partenza la corrente è alta: non conta
const byte RIPOSO_X = 9;             // riposo = 9 volte il tempo di moto (10 %)

struct Pulsante {
  byte pin;
  bool premuto;               // stato stabile
  bool letto;                 // ultima lettura
  unsigned long cambio;       // quando la lettura è cambiata
};
Pulsante apri = {P_APRI, false, false, 0};
Pulsante chiudi = {P_CHIUDI, false, false, 0};
```

```
enum Stato { FERMO, APRE, CHIUDE, RIPOSA };
Stato stato = FERMO;
char posizione = '?';         // A aperto, C chiuso, M a metà, ? non si sa
unsigned long inizio;         // partenza del movimento o del riposo
unsigned long riposo;         // quanto deve durare il riposo
unsigned long normale;        // ultimo istante con la corrente sotto soglia
int picco;                    // corrente più alta vista dopo lo spunto

bool tocco(Pulsante &p) {     // vero una volta sola, quando il pulsante viene premuto
  bool ora = digitalRead(p.pin) == LOW;
  if (ora != p.letto) {
    p.letto = ora;
    p.cambio = millis();
  }
  if (millis() - p.cambio > 30 && p.premuto != p.letto) {   // stabile da 30 ms
    p.premuto = p.letto;
    return p.premuto;
  }
  return false;
}

int correnteMA() {
  long somma = 0;
  for (byte i = 0; i < 20; i++) somma += analogRead(SENSE);
  float mV = somma / 20.0 * 1100.0 / 1024.0;
  return (int)(mV * 5);
}

void parti(Stato s) {
  stato = s;
  digitalWrite(SLP, HIGH);
  delay(2);
  digitalWrite(AIN1, s == APRE ? HIGH : LOW);
  digitalWrite(AIN2, s == CHIUDE ? HIGH : LOW);
  inizio = normale = millis();
  picco = 0;
  Serial.println(s == APRE ? "Apro" : "Chiudo");
}

void ferma(const char *motivo, char dove) {
  digitalWrite(AIN1, LOW);
  digitalWrite(AIN2, LOW);
  digitalWrite(SLP, LOW);
  unsigned long durata = millis() - inizio;
  posizione = dove;
  riposo = durata * RIPOSO_X;
  inizio = millis();
  stato = RIPOSA;
  char riga[72];
  snprintf(riga, sizeof riga, "Fermo (%s) dopo %lu ms, picco %d mA, riposo %lu s",
           motivo, durata, picco, riposo / 1000);
  Serial.println(riga);
}

void setup() {
  pinMode(P_APRI, INPUT_PULLUP);
  pinMode(P_CHIUDI, INPUT_PULLUP);
```

```
  pinMode(AIN1, OUTPUT);
  pinMode(AIN2, OUTPUT);
  pinMode(SLP, OUTPUT);
  pinMode(LED_BUILTIN, OUTPUT);
  Serial.begin(115200);
  analogReference(INTERNAL);
  for (byte i = 0; i < 5; i++) analogRead(SENSE);
  Serial.println("Pronto");
}

void loop() {
  bool a = tocco(apri);
  bool c = tocco(chiudi);
  unsigned long adesso = millis();
  if (stato == APRE || stato == CHIUDE) {
    char fine = stato == APRE ? 'A' : 'C';
    unsigned long t = adesso - inizio;
    int mA = correnteMA();
    if (t > SPUNTO_MS && mA > picco) picco = mA;
    if (t < SPUNTO_MS || mA < SOGLIA_MA) normale = adesso;
    if (a || c) ferma("pulsante", 'M');
    else if (adesso - normale > 150) ferma("corrente alta", fine);
    else if (t > TEMPO_MAX) ferma("tempo massimo", fine);
  } else if (stato == RIPOSA) {
    digitalWrite(LED_BUILTIN, (adesso / 250) % 2 ? HIGH : LOW);   // il LED L lampeggia
    if (adesso - inizio > riposo) {
      stato = FERMO;
      digitalWrite(LED_BUILTIN, LOW);
      Serial.println("Pronto");
    }
  } else if (a && posizione != 'A') {
    parti(APRE);
  } else if (c && posizione != 'C') {
    parti(CHIUDE);
  }
}
```

8 Regola la soglia

`SOGLIA_MA = 800` è un valore di partenza nostro. Monta lo sportello (passo 9), poi aprilo e fermalo a metà con un pulsante: il «picco» che il programma scrive in quella riga è la corrente più alta della corsa, con il carico vero. Metti la soglia a metà strada tra quel picco e la corrente che hai visto in fondo nella prova del passo 6. Se nella prova in fondo la corrente crollava a zero, la soglia serve solo per gli ostacoli: tienila un po' sopra il picco.

> **✓ Cosa devi vedere**
> Premi APRI: lo sportello si apre e si ferma da solo, il LED L lampeggia, poi il monitor scrive «Pronto». Premi CHIUDI: si richiude.

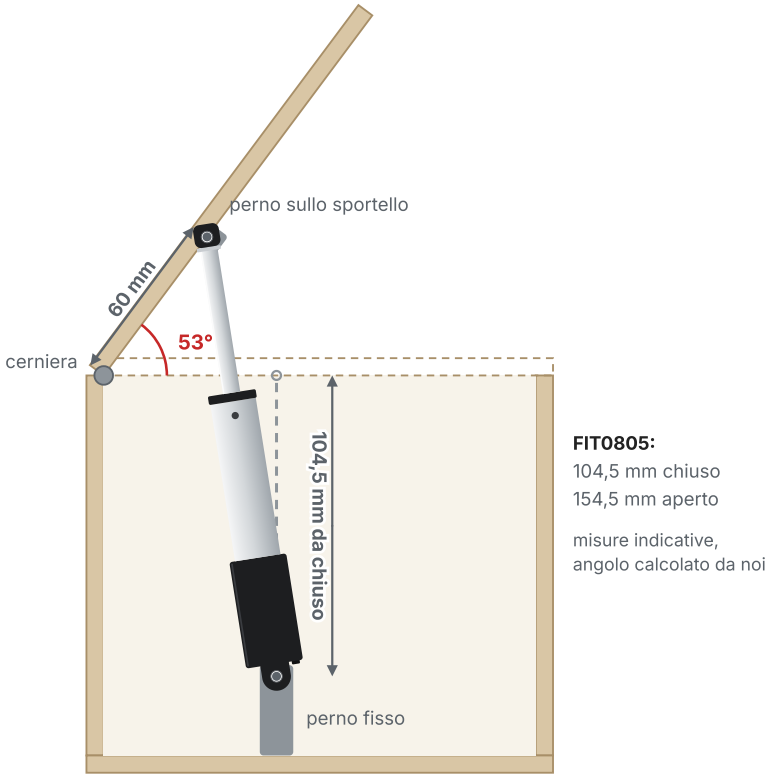
Se non funziona

- Si ferma subito dopo la partenza con «corrente alta»: per il tuo carico la soglia è bassa, alzala.
- Arriva in fondo e ronza fino al «tempo massimo»: la soglia è alta, abbassala restando sopra il picco della corsa.
- I pulsanti non rispondono: sta riposando (LED L che lampeggia), oppure lo sportello risulta già aperto o già chiuso e serve l'altro pulsante.

9 Montarlo sullo sportello

L'attuatore deve poter ruotare ai due capi: un perno nell'orecchia, fissato alla scatola, e uno nell'occhiello dello stelo, fissato allo sportello. Per i fori da 4 mm una vite M3 con dado autobloccante, stretta senza bloccare, fa da perno (proposta nostra). Così lo stelo spinge solo lungo il suo asse, senza forze di traverso.

Uno sportello a ribalta: i due perni e l'angolo di apertura



informaticait.it

Nel disegno la cerniera è a sinistra, il perno sullo sportello a 60 mm da lei e il perno fisso 104,5 mm più in basso, cioè l'interasse da chiuso del FIT0805. Quando lo stelo esce di 50 mm lo sportello si apre di circa 53° (calcolo nostro, misure indicative). Con il perno più vicino alla cerniera lo sportello si apre di più, ma serve più forza.

DFRobot non dice se lo stelo tiene la posizione quando il motore è spento e un peso lo spinge: provalo con il tuo sportello prima di fidarti.

Attenzione

- Lo sportello si chiude con una spinta che può arrivare a 128 N: tieni le dita lontane dal bordo. Il limite di corrente e la soglia aiutano, ma non sono un dispositivo di sicurezza.

Prova tu

- Collega il pin FLT del driver a un pin libero con `INPUT_PULLUP` e scrivi «guasto» quando va basso: TI lo abbassa per sovracorrente e surriscaldamento.
- Una presa d'aria che si apre sopra i 28 gradi, con il TMP36 della guida n. 40.
- Un secondo attuatore sul ponte B: BIN1, BIN2, BOUT1, BOUT2 e BS per la corrente.

✓ Cosa hai imparato

- l'attuatore cambia verso se scambi i due fili: serve un ponte H;
- se la corrente del motore non è dichiarata, scegli un driver che la limiti da solo;
- la tensione su una resistenza da 0,2 Ω dice quanta corrente passa, e quando lo stelo è bloccato;
- il 10 % dell'etichetta, letto come ciclo di lavoro, vuol dire muoversi poco e riposare tanto.

Trovi i componenti su informaticait.it

- Attuatore lineare elettrico (pistone) 6 V, corsa 10 mm, spinta 128 N, IP54
- Attuatore lineare elettrico (pistone) 6 V, corsa 30 mm, spinta 128 N
- Attuatore lineare elettrico (pistone) 6 V, corsa 50 mm, spinta 128 N
- Attuatore lineare elettrico (pistone) 6 V, corsa 100 mm, spinta 128 N
- Driver per motori DRV8833 a doppio ponte H per 2 motori DC o 1…
- Scheda Arduino Uno Rev3 originale
- Portabatterie 4 × AA con coperchio a scatto

Clicca sul nome per aprire la scheda prodotto. Disponibilità e prezzi aggiornati sul sito.

Fonti: DFRobot, attuatori FIT0803, FIT0804, FIT0805 e FIT0806: pagine prodotto, disegno quotato e foto · Texas Instruments, DRV8833 datasheet (SLVSAR1E) · Adafruit, DRV8833 DC/Stepper Motor Driver Breakout (3297): pagina prodotto, guida Learn e file Eagle · Toshiba, TB6612FNG datasheet; Adafruit, pagine prodotto 3190 e 4489 · Adafruit, portapile 4 × AA con interruttore (830) · Arduino, riferimento di analogReference() e pinMode() · Guide informaticait n. 8, 35, 40 e 49