Arduino – ESC e motori droni
Speciale «Festivolare 2017»

Versione 1.0
18/03/2020

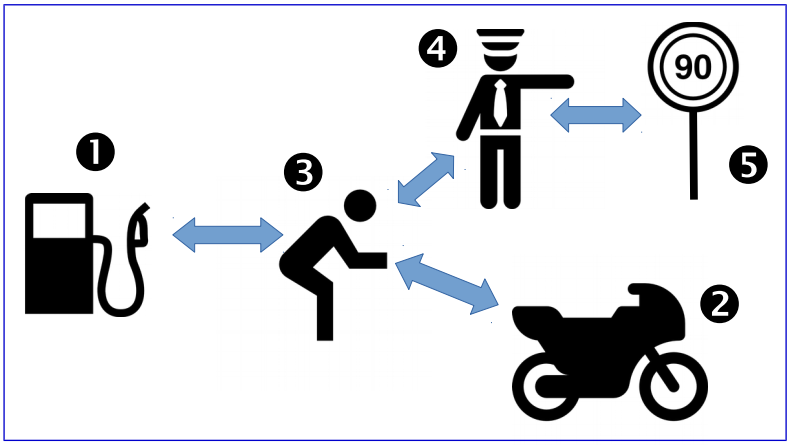
In questo tutorial impareremo quali sono i componenti fondamentali utilizzati in tutti i droni ed i modellini di aerei che sfruttano motori elettrici per la propulsione; vedremo come collegarli e come controllare la velocità del motore grazie ad uno sketch di Arduino.
Puoi trovare tutto il materiale (tutorial, sketch e schemi) qui: **coderdojotrento.it/arduino3**

N.B. Data l'intensità della corrente che può essere erogata della batterie e data la potenza che caratterizza questi motori, si raccomanda di prestare molta attenzione durante la preparazione del circuito; se poi si intende anche effettuare qualche prova dopo aver montato delle eliche, l'utilizzo degli occhiali di protezione è obbligatorio.

Gli attori in gioco

Per cominciare elenchiamo quello che ci serve, prendendo spunto dal fumetto qui a destra:

❶ una sorgente di energia,
❷ un motore,
❸ un pilota che riesca sfruttare l'energia per far lavorare il motore,
❹ un controllore che a sua volta tenga d'occhio il pilota spericolato e gli indichi che velocità tenere (❺)



Vediamo ogni singolo elemento del disegno, per capire cosa rappresenta nel nostro circuito: ci aiuterà molto per comprendere come dovranno essere collegati tra loro i vari componenti.

Batterie. La nostra fonte di energia sono delle batterie costruite appositamente per alimentare i motori elettrici: sono batterie Li-po ricaricabili, e sono in grado di fornire una grossa quantità di corrente, dato che i motori utilizzati nei droni possono sviluppare un'elevata potenza. Come puoi vedere dall'immagine qui a destra, sono caratterizzate da molti parametri (guarda quanti numeri compaiono sulla confezione!), ma i principali sono voltaggio e carica immagazzinata. Sono poi quasi sempre presenti due fasci

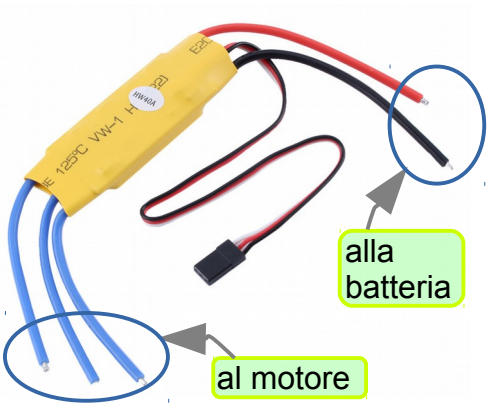
Trovi questo ed altri tutorial su coderdojotrento.it
Realizzato con il supporto di CoderDolomiti.
Licenza: CC-BY Alessandro Ebranati, CoderDojo Trento
Quest'opera è distribuita con Licenza Creative Commons Attribuzione 4.0 Unported

di cavetti: una serve per prelevare la corrente, l'altra per ricaricare la batteria.

Motore.

Il *motore brushless* ("senza spazzole") è il componente che -grazie ai magneti permanenti ed alle bobine di cui è composto- riesce a trasformare l'energia elettrica in movimento: moltissimi sono i modelli in commercio, come pure sono molti gli accessori che li accompagnano (staffe e viti per fissarlo alla struttura, fermi per le eliche, …). Notare i tre fili: chi può essere in grado di attivare nel momento giusto le diverse bobine, in modo da far ruotare a piacimento il rotore? Andiamo a conoscerlo.

ESC.



Il "*controllore elettronico della velocità*" (in inglese: *"electronic speed control"* o ESC, per brevità) ha il ruolo del pilota esperto: sfrutta l'energia a sua disposizione (che arriva grazie ai due cavi rosso e nero) per far girare a piacimento il motore, attivando in alternanza i tre cavetti azzurri che vanno collegati alle bobine del motore. I circuiti dentro l'ESC sono infatti in grado di trasformare la corrente continua fornita dalla batteria in tre segnali alternati che permettono di orientare il rotore (la parte mobile del motore) come vogliamo.

Non è finita: oltre ai cinque cavi di grande diametro che permettono di far passare l'elevata corrente che serve al motore, ci sono tre piccoli cavetti che useremo per il collegamento col prossimo componente.

Centralina per il controllo della velocità.

Come abbiamo detto, il pilota (l'ESC) è molto bravo, ma è anche un po' indisciplinato: dobbiamo quindi abbinarlo ad un altro elemento che gli dica sempre a che velocità andare.

È facile riconoscere la centralina che abbiamo scelto qui a destra: si tratta di un Arduino! Qui starà tutta l'intelligenza del nostro sistema (lo sketch, ovvero il programma che possiamo lanciare su Arduino).

Selettore di velocità.

Lo strumento che useremo per decidere a quale velocità far girare il motore sarà un semplice potenziometro angolare: sappiamo già che possiamo collegarlo ad uno dei canali di input analogici di Arduino e leggere il valore corrispondente. In questo modo il potenziometro diventerà in pratica la nostra manopola del gas!

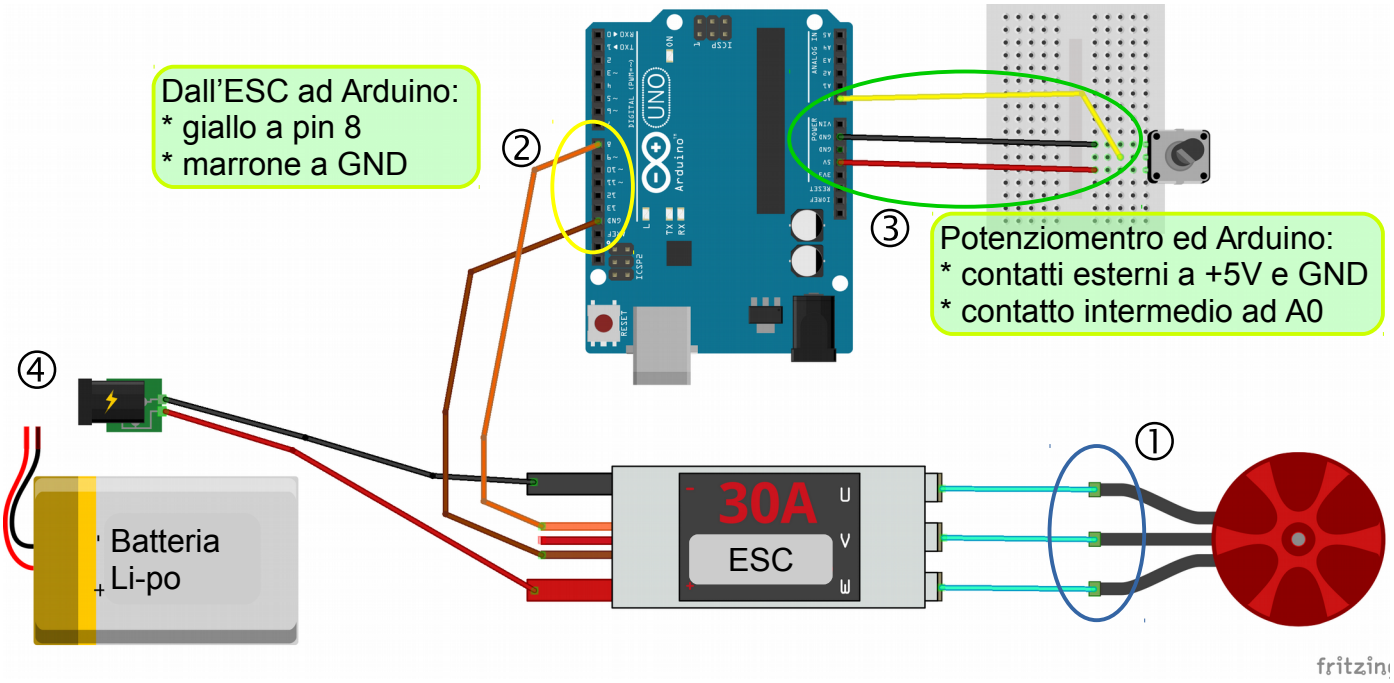
Vediamo ora lo schema di collegamento dei diversi componenti: alcuni dei cavi hanno i terminali dotati di particolari connettori che sono pensati apposta per facilitare la costruzione del circuito. Bisogna comunque prestare molta attenzione, dato che la corrente fornita dalla batteria potrebbe facilmente rovinare i componenti elettronici. Per questo **prepareremo il circuito, lasciando scollegata la batteria**: la collegheremo solo alla fine, dopo aver testato per bene tutto il resto.

Se hai dei dubbi, chiedi al mentor di dare un'occhiata al tuo lavoro.

Trovi questo ed altri tutorial su coderdojotrento.it
Realizzato con il supporto di CoderDolomiti.
Licenza: CC-BY Alessandro Ebranati, CoderDojo Trento
Quest'opera è distribuita con Licenza Creative Commons Attribuzione 4.0 Unported

I collegamenti

Questo lo schema finale dei collegamenti: vediamo l'immagine completa, e analizziamo i singoli passi da eseguire nell'ordine in cui vengono presentati. N.B. Per ora la batteria rimane scollegata.



① Fissiamo il motore ad un supporto stabile utilizzando le viti e le staffe fornite: ricordiamoci che il rotore (la parte mobile) è quella esterna, quindi non potremmo bloccarlo semplicemente tenendolo in mano! Colleghiamo i tre cavi del motore a quelli azzurri dell'ESC: l'ordine in cui li attacchiamo alle uscite denominate U, V e W determina il verso di rotazione del motore, ma questo non è importante finché non fissiamo un'elica all'asse del motore.

② Il collegamento tra Arduino (la nostra centralina) e l'ESC si effettua tramite i cavetti sottili che escono in un fascio di tre; di solito i colori sono marrone, rosso ed arancione (come nel disegno) oppure nero, rosso e giallo: il terzo cavetto (arancione o giallo) è quello che Arduino usa per inviare il segnale di controllo all'ESC e noi lo collegheremo al pin 8 di Arduino.

BEC ("Battery-eliminator circuit" o "Circuito per l'eliminazione della batteria")
Gli altri due cavetti (quello marrone/nero e quello rosso) hanno i colori tipici dell'alimentazione ed infatti servono proprio a questo: ricordiamo che l'ESC verrà attaccato ad una batteria molto capace, quindi -volendo- potremmo utilizzarli per fornire la corrente ad Arduino (eliminando quindi il bisogno di una seconda batteria per alimentare la centralina). Nel nostro caso terremo Arduino collegato alla porta USB del computer e siccome questa fornisce anche l'alimentazione, quei due cavetti non servono.

Ricordiamoci però che ogni circuito deve essere "chiuso", quindi dobbiamo avere il *cavetto di ritorno* per quello arancione che fornisce il segnale da Arduino all'ESC: per questo motivo colleghiamo il cavetto marrone/nero dell'ESC al GND di Arduino.

③ Per usare il potenziometro angolare è sufficiente portare sui due estremi la tensione di riferimento (collegandoli ai 5V ed a GND di Arduino) e collegare il piedino di mezzo al canale A0.

④ Come già detto, la batteria rimane per ora ancora scollegata: potremmo attaccarla solo dopo aver sistemato la logica di controllo, resettato i componenti ed aver effettuato qualche test.

Licenza: CC-BY Alessandro Ebranati, CoderDojo Trento
Quest'opera è distribuita con Licenza Creative Commons Attribuzione 4.0 Unported

La logica di controllo

Questo è lo sketch che useremo nella nostra centralina: trascriviamolo nell'IDE di Arduino e poi commentiamo i principali passaggi. (qui trovi il codice, per scaricare fai click destro –> salva come)

```
1  #include <Servo.h>                // utilizziamo la stessa libreria dei servo-motori
2
3  ////// BLOCCO DI DEFINIZIONE ED INIZIALIZZAZIONE DELLE VARIABILI //////
4  Servo ESC;                        // riferimento ad un servo-motore di nome 'ESC'
5  const int pinESC = 8;             // costante per il pin al quale è collegato l'ESC
6  int valESC;                       // variabile che contiene il valore che manderemo all'ESC
7  const int pinPot = A0;            // costante per il pin al quale è collegato il potenziometro
8  int valPot;                       // variabile che contiene il valore letto dal potenziometro
9
10 ////// METODO DI INIZIALIZZAZIONE //////
11 void setup() {
12   ESC.attach(pinESC);             // agganciamo l'ESC al pin numero 8
13   ESC.writeMicroseconds(1000);    // mandiamogli il segnale di fermo
14   Serial.begin(9600);             // inizializziamo la comunicazione col monitor seriale
15 }
16
17 ////// METODO ESEGUITO CONTINUAMENTE //////
18 void loop() {
19   valPot = analogRead(pinPot);               // leggo il valore del potenziometro
20   valESC = map(valPot, 0, 1023, 1000, 2000); // lo trasformo nell'intervallo da 1000 a 2000
21   Serial.println(valESC);                    // scrivo il valore calcolato sulla seriale ...
22   ESC.writeMicroseconds(valESC);             // ... e lo invio anche all'ESC
23   delay(20);                                 // attende un cinquantesimo di secondo (20 millisec)
24 }
```

① Il motore brushless è completamente diverso dai servomotori che già conosciamo, ma può essere comandato per mezzo della stessa libreria: a riga 1 includiamo quindi l'header file *Servo.h*.

② Nella sezione di definizione della variabili che useremo nel programma (righe da 4 a 8) prepariamo il riferimento ad una classe di tipo Servo che ci permetterà di sfruttare le funzionalità messe a disposizione dalla libreria e la chiamiamo ESC. Le altre due coppie di riferimenti sono la costante per il pin e la variabile che useremo per il segnale di controllo dell'ESC e per il potenziometro (collegato al pin A0, il primo canale analogico di input).

③ Con le righe 12 e 13 del metodo *setup()* indichiamo che il segnale dell'ESC è agganciato al pin 8 e tramite la funzione *writeMicroseconds* chiediamo di tenere la velocità minima (di solito gli ESC accettano in input un segnale ad impulso con una larghezza che va da 1000 a 2000 microsecondi), ossia partiamo con il motore fermo. A riga 14 apriamo poi il dialogo tra Arduino e computer utilizzando il cavo USB per una una comunicazione seriale: la velocità di trasmissione è quella solita di 9600 baud.

④ Nel metodo loop() che viene continuamente eseguito procediamo così: leggiamo il valore settato dal potenziometro (dall'*analogRead* invocato a riga 19 otteniamo valori che vanno da un minimo di 0 ad un massimo di 1023), grazie alla funzione *map()* lo trasformiamo in un valore che va da 1000 a 2000 (riga 20) per poterlo usare nella funzione *writeMicroseconds* dell'ESC (riga 22); tracciamo il valore calcolato scrivendolo sulla seriale (riga 21) e quindi attendiamo un poco prima di eseguire un'altra volta il metodo *loop()*.

Trovi questo ed altri tutorial su coderdojotrento.it
Realizzato con il supporto di CoderDolomiti.
Licenza: CC-BY Alessandro Ebranati, CoderDojo Trento
Quest'opera è distribuita con Licenza Creative Commons Attribuzione 4.0 Unported

Gli ultimi test ...

Verifichiamo per prima cosa il codice, compilandolo e caricandolo su Arduino: se non abbiamo commesso errori, dovremmo quindi essere in grado di aprire il "*Monitor seriale*", settare la velocità a 9600 baud e leggere quindi i valori della variabile valEsc che Arduino invia al computer ad ogni esecuzione del metodo l*oop()* (vedi istruzione di riga 21).

Ruota lentamente la manopola del potenziometro da un estremo all'altro: quando il cursore è spostato verso il contatto di terra (il filo nero che abbiamo collegato a GND) dobbiamo leggere il valore 1000 (che corrisponde allo 0 del potenziometro); quando la manopola è invece girata tutta verso il contatto a +5 Volt (il cavetto rosso) il valore letto sul monitor dovrà essere 2000 (che corrisponde al valore 1023, il massimo letto dal potenziometro).

Il programma caricato sulla nostra centralina di controllo sembra a posto!

… e siamo pronti per partire!

Leggi attentamente la sequenza di istruzioni necessarie per attivare il motore; solo dopo potrai effettivamente eseguire i passi che garantiscono una "prova di volo" ben riuscita.

① Fai partire Arduino ed apri la finestra del "Monitor seriale": assicurati che il programma stia girando, provando a ruotare la manopola del potenziometro e verificando che vengano correttamente prodotti dei valori che stanno tra 1000 e 2000.

② Porta il potenziometro nella posizione di zero: deve essere ruotato tutto verso il contatto di terra e di conseguenza il valore letto sul monitor della seriale deve essere 1000.

Collega la batteria Li-po all'ESC; se tutte le altre connessioni sono state fatte in maniera corretta, sentirai un motivetto *provenire dal motore*: l'unico modo che l'ESC ha per comunicarci qualcosa -infatti- è far vibrare il motore ad una frequenza tale che può emettere suoni!
Ecco quello che dovremmo sentire:

1. una sequenza beep-beep-beep crescente,
2. una pausa,
3. due oppure tre beep (in relazione alla batteria che abbiamo collegato, che sarà 2C o 3C),
4. una pausa,
5. un lungo beep finale.

A questo punto l'ESC è pronto. Se invece senti una serie continua di beep, significa che qualche collegamento non è stato fatto in maniera corretta, oppure che il segnale che arriva da Arduino non è comprensibile per l'ESC (rivedi quindi il circuito e lo sketch assieme al mentor).

③ Se l'ESC è "contento", possiamo cominciare a ruotare **molto lentamente** la manopola del potenziometro. Appena il segnale avrà superato una soglia minima (attorno a 1100), il motore comincerà a girare: è abbastanza rumoroso, te ne accorgerai di sicuro!

Prova a ruotare il potenziometro per avvicinarti alla velocità massima, ma torna rapidamente ad un basso numero di giri. Non far girare comunque il motore al massimo, perché abbiamo un elevato consumo di corrente e di sicuro stiamo disturbando i nostri vicini: stiamo solo facendo un esperimento con qualche componente di elettronica, non un vero volo o una gara di Formula 1!

④ Riporta la manopola al valore minimo e -una volta che il motore sarà fermo- **stacca la batteria che alimenta l'ESC facendo attenzione a non tirare i fili.** Et voilà, atterrati! ☺

Cosa abbiamo imparato?

Trovi questo ed altri tutorial su coderdojotrento.it
Realizzato con il supporto di CoderDolomiti.
Licenza: CC-BY Alessandro Ebranati, CoderDojo Trento
Quest'opera è distribuita con Licenza Creative Commons Attribuzione 4.0 Unported

Di sicuro abbiamo imparato che siamo in grado di utilizzare anche componenti “da grandi”, che vanno trattati con particolare attenzione, a patto di fare i nostri esperimenti con la supervisione di un adulto. Inoltre abbiamo capito che possiamo portare a termine anche dei progetti complicati, se prima pianifichiamo bene le operazioni da svolgere e facciamo anche qualche test preliminare.

Se non progettiamo bene il nostro lavoro, c’è infatti il rischio di avere brutte sorprese!

Ringraziamenti

Questo tutorial:
* fa parte delle “risorse” del CoderDojo Trento,
* è stato scritto con il supporto di CoderDolomiti,
* ma soprattutto: è approvato da “Violet the pilot”!

Si ringraziano per l’ospitalità

Un grazie particolare a Steve Breen, per la concessione di utilizzo delle immagini tratte dal suo libro “Violet the pilot” (ISBN-13: 978-0803731257) pubblicato da Dial Books.

Crediti per le altre immagini utilizzate nel tutorial.

Le icone sono state disegnate da Plainicon per Flaticon (Licenza CC 3.0 BY); per lo schema del circuito di pag. 3 abbiamo usato fritzing

Licenza: CC-BY Alessandro Ebranati, CoderDojo Trento
Quest'opera è distribuita con Licenza Creative Commons Attribuzione 4.0 Unported