Dispense su Arduino e Robotica pubblicate su Internet

Volevo condividere con voi un programma C# ed un Firmware Arduino che ho scritto per calcolare la posizione in una traiettoria punto-punto imponendo velocità iniziale e finale zero.
Ho utilizzato il polinomio cubico a0 + a1*t + a2*(t^2) + a3*(t^3) che permette una variazione "dolce" della posizione e della velocità, essenziale per non avere bruschi movimenti che potrebbero danneggiare i motori e causare vibrazioni.

Il programma C# l'ho scritto per provare l'algoritmo e generare i dati da graficare in Excel, il firmware Arduino è stato testato con un paio di servo low-cost (HD-1800A ed MG995), i risultati sono fortemente influenzati dal fatto che i servo in oggetto hanno un loro sistema di controllo analogico basato su potenzimetro per cercare la posizione corretta. Nelle prove si passeranno i valori della posizione angolare che i servo dovranno raggiungere per passare da qo=0 a qf=180 gradi in un tempo tf=400ms.

Programma C#

Codice:

```
static void Traiettoria()
  {
   File.Delete("res.csv");
   // Utilizzo polinomio cubico
   // a0 + a1*t + a2*(t^2) + a3*(t^3)
   // per calcolare la posizione ed imporre velocità iniziale e finale = 0
   double tf = 400;          // tempo finale
   double q0 = 0;            // posizione iniziale
   double qf = 180;          // posizione finale
   double qf0 = qf - q0;
   double tf2 = Math.Pow(tf, 2);
   double tf3 = Math.Pow(tf, 3);
   string csv = string.Empty;    // stringa per generare il file .csv

   for (double t = 1; t < tf + 1; t += 1)
   {
    double t2 = Math.Pow(t, 2);
    double t3 = Math.Pow(t, 3);

    // Calcolo coefficienti eq. polinomiale
    double a0 = q0;
    // a1 = 0, velocità iniziale = 0
    double a2 = (3 * qf0) / tf2;
    double a3 = (-2 * qf0) / tf3;

    double p = a0 + (a2 * t2) + (a3 * t3);    // posizione
    double v = (2 * a2 * t) + (3 * a3 * t2);  // velocità
    double a = (2 * a2) + (6 * a3 * t);       // accelerazione
```

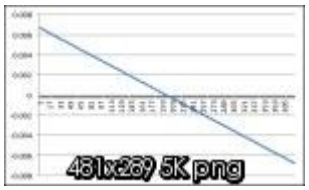
```
    csv += p + ";" + v + ";" + a + ";" + "\r\n";
   }

   File.WriteAllText("res.csv", csv);   // scrivi il file con i risultati
   Process.Start("res.csv");            // apri il file (Excel)
  }
```

L'algoritmo viene valutato ogni unità di tempo, ecco i profili che si ottengono, grafici generati con Excel:

Posizione

Velocità

Accelerazione



Come si può vedere l'unico problema potrebbe essere l'accelerazione, si potrebbe dire che il motore "parte in quinta" anche se nella realtà l'accelerazione sarà una rampa più o meno ripida date le caratteristiche fisiche del motore.
Se siete interessati ad una trattazione più matematica dell'algoritmo e dei suoi parametri fatemi sapere che pubblico qualche info aggiuntiva.

Firmware Arduino

Codice:

```
// Utilizzo polinomio cubico
// a0 + a1*t + a2*(t^2) + a3*(t^3)
// per calcolare la posizione ed imporre velocità iniziale e finale = 0
#include <Servo.h>
Servo servo;

const float tf = 400.0f;      // tempo finale
```

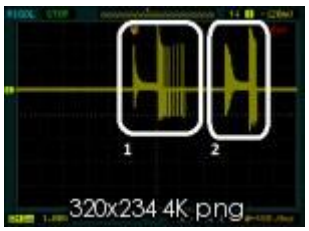
```
const float q0 = 0.0f;       // posizione iniziale
const float qf = 170.0f;      // posizione finale
const float qf0 = qf - q0;
const float tf2 = pow(tf, 2);
const float tf3 = pow(tf, 3);

unsigned long startTime, time;
void setup()
{
  servo.attach(9);  // servo sul pin9
  servo.write(0);   // vai a 0°
  delay(1000);      // aspetta il posizionamento
  startTime = millis();  // inizia a contare il tempo da ora
}

void loop()
{
  time = millis() -  startTime;
  servo.write(GetPosition(time));
}

float GetPosition(long t)
{
  if(t > tf)
  {          // Se siamo oltre il tempo finale
    return qf; // ritorna la posizione finale
  }

  // Calcola il quadrato e il cubo del tempo
  float t2 = pow(t, 2);
  float t3 = pow(t, 3);

  // Calcolo coefficienti eq. polinomiale
  float a0 = q0;
  // a1 = 0, velocità iniziale = 0
  float a2 = (3 * qf0) / tf2;
  float a3 = (-2 * qf0) / tf3;

  // Calcolo risultati
  float p = a0 + (a2 * t2) + (a3 * t3);       // posizione
  //double v = (2 * a2 * t) + (3 * a3 * t2);   // velocità
  //double a = (2 * a2) + (6 * a3 * t);       // accelerazione

  return p;
}
```

Ho effettuato delle prove visive, in effetti si nota che la velocità segue il profilo prima graficato, e delle prove di consumo energetico con una resistenza Shunt di 1.2 ohm e 6V di tensione.
Nelle immagini 1 è il movimento del servo "grezzo" da 180 a 0°, mentre 2 è il movimento tramite la traiettoria punto-punto.
Si nota subito una maggiore facilità a trovare il punto di arrivo da parte del controllo del servo ed un consumo energetico di picco minore oltre a meno variazioni, col vantaggio di emettere meno rumore elettromagnetico.

Servo da 9g HD-1800A http://www.hobbytronics.co.uk/datasheets/HD-1800A.pdf

Ho invertito i morsetti dell'oscilloscopio quindi il grafico risulta ribaltato, è chiaro però che torna quanto detto

Servo Standard MG995



Questo servo è "famoso" per far fatica (senza carico) a trovare il punto di arrivo, con diverse oscillazioni attorno al punto finale. Il posizionamento tramite la traiettoria mostra che il servo trova con più facilità il punto di arrivo.

Concludendo, seppur la traiettoria punto punto implementata tramite equazione polinomiale sia maggiormente efficace su motori DC, anche tramite i servo è possibile sperimentare qualcosa e si può avere qualche vantaggio.

UPDATE: Utilizzando servo.writeMicroseconds() al posto di servo.write() è possibile ottenere una risoluzione quasi 8 volte maggiore (0.09° invece di 0.7°) e di conseguenza movimenti meno scattosi.