ROBOFUN.RO

LECȚIA XVIII

CURS GRATUIT

ARDUINO ȘI ROBOTICĂ

Motoare Pas cu Pas

Arduino, motoare stepper si Easydriver

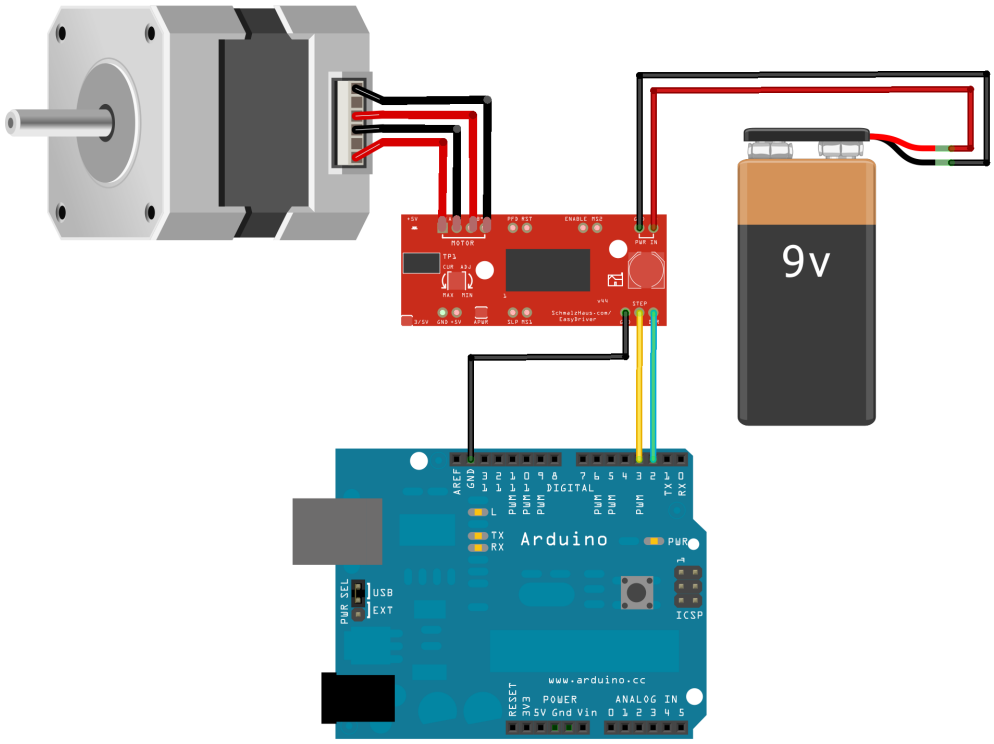
Motoarele pas cu pas sau motoarele stepper sunt motoare de curent continuu, fara perii si sunt ideale daca vrei sa le integrezi intr-o anumita aplicatie, ce necesita o anumita viteza de rotatie sau daca vrei ca motorul sa se roteasca pana intr-un anumit punct si apoi sa isi pastreze pozitia.

Un motor de curent continuu poate fi controlat in sensul miscarii intr-un anumit sens cu o viteza data, lucru pe care il faci prin aplicarea unei anume tensiune la bornele sale. Motorul se roteste cat timp exista tensiune aplicata. Nu vei putea insa sa ii controlezi exact rotatia (spre exemplu, nu ai cum sa-l rotesti cu fix 45 de grade, si apoi sa il opresti).

Modul de functionare al motoarelor pas cu pas este diferit. O rotatie completa a unui motor stepper este alcatuita din mai multi pasi, fiecare pas reprezentand doar o fractiune din rotatie completa a motorului. Lucrul asta se datoreaza constructiei interne, rotorul fiind compus din magneti permanenti, iar statorul din infasurari. Din acest motiv, un motor pas cu pas poate fi controlat extrem de precis. Il poti roti spre exemplu cu 1 grad spre stanga (adica a 360-a parte dintr-o rotatie completa a axului). Sau il poti roti cu 45 de grade spre dreapta si apoi il poti bloca.

Sigur, si controlul este ceva mai complicat decat in cazul unui motor de curent continuu. Din fericire, folosind Arduino si un driver specializat, lucrurile devin simple. In cele ce urmeaza vom prezenta folosirea EasyDriver (un driver de motor pas cu pas specializat) pentru a controla un motor pas cu pas.

Diagrama de conectare:



Arduino PWM 3	EasyDriver STEP
Arduino PWM 2	EasyDriver DIR
Aduino GND	EasyDriver GND

EasyDriver iti permite sa comanzi motorul in pasi foarte mici. Aceasta tehnica se numeste microstepping, practic driverul imparte un pas in 8 micropasi. Motoarele de uz general realizeaza o rotatie completa in 200 de pasi sau unghiul unui pas este de 1,8°. Dar pentru ca EasyDriver imparte un pas in 8 micropasi atunci sunt necesari 1600 de micropasi pentru o rotatie completa a motorului. Asta inseamna ca motorul se poate roti cu precizie ridicata. La viteze mari, motorul dezvolta insa o forta redusa.

Inainte de a pune in functiune circuitul trebuie sa fii atent cand alegi sursa de tensiune, pentru ca trebuie sa respecti parametrii motorului: tensiunea de alimentare, si consum. In general, motoarele se alimenteaza la 12 V. Spre exemplu, si aceasta - http://www.robofun.ro/stepper_motor_100g_cm si acesta - http://www.robofun.ro/motor_stepper si acesta – http://www.robofun.ro/motor_stepper_400_pasi_4_8_kg_cm, toate functioneaza la 12 V. Celalalt parametru important este intensitatea curentului necesar (eista o relatie directa intre aceasta si forta motorului; cu cat motorul necesita un curent mai mare pentru a functiona, cu atat trebuie sa te astepti ca va avea o forta mai mare).

Inainte de a trece la cod, vreau sa te pun in garda asupra faptului ca un motor pas cu pas se va

incalzi in timpul functionarii, iar acest lucru este normal. Pentru a functiona, prin bobinele motorului trece curent electric tot timpul, chiar si cand motor sta pe loc (in regim de blocare). Cat timp poti atinge motorul cu mana, este OK (60 – 80 de grade Celsius). Vezi si discutia de aici - http://robofun.ro/forum/viewtopic.php?f=4&t=139&p=478&hilit=caldura#p478

Daca vrei sa tii motorul mai rece, ai doua solutii. Prima ar fi sa-i reduci curentul. Pe placa EasyDriver, exista un mic potentiometru din care poti face acest lucru. Evident, cu cat curentul este mai mic, cu atat forta motorului este mai mica. Al doilea lucru pe care il poti face este ca atunci cand motorul sta, sa faci disable la driver. Pentru a realiza acest lucru, conecteaza pinul *ENABLE* de pe placa EasyDriver la un pin digital Arduino. Atunci cand cobori in zero pinul digital, prin motor nu va mai circula curent (motorul se va raci). Evident, motorul va fi liber in acest moment sa se invarta (daca exista forte externe in sistemul tau mecanic). De la caz la caz, poti sau nu a aplici aceasta abordare. Spre exemplu, pentru un robot pasitor umanoid, nu poti face asta, pentru ca in momentul in care faci disable la driver, robotul tau va cadea la pamant. Daca insa este vorba despre un sistem care misca o draperie, atunci nu este nici o problema daca dupa ce am terminat miscarea draperiei fac disable la driver. Draperia va ramane in pozitia curenta, iar motorul nu va mai consuma curent si nu se va mai incalzi.

Codul sursa

Primele linii de cod se evidentiaza prin doua directive:

```
#define DIR_PIN 2
#define STEP_PIN 3
```

Oriunde este scris in cod DIR_PIN, programul se refera la pinul 2 de la Arduino. Acelasi lucru este valabil si pentru STEP_PIN.

```
void setup() {
  pinMode(DIR_PIN, OUTPUT);
  pinMode(STEP_PIN, OUTPUT);
}
```

Nivelele logice, impulsurile sau bitii, depinde cum preferi sa ii numesti circula dinspre Arduino spre EasyDriver asa ca DIR_PIN si STEP_PIN vor fi setati ca iesire.

```
void loop(){
  //rotate a specific number of degrees
  rotateDeg(360, 1);
  delay(1000);
  rotateDeg(-360, 0.1);  //reverse
  delay(1000);
  //rotate a specific number of microsteps (8 microsteps per step)
  //a 200 step stepper would take 1600 micro steps for one full
//revolution
  rotate(1600, 0.5);
  delay(1000);
  rotate(-1600, 0.25); //reverse
  delay(1000);
}
```

Loop este o bucla repetitiva, executa consecutiv liniile de cod din interiorul acesteia. In bucla vei observa ca se apeleaza functiile rotateDeg, rotate si delay. Functia delay accepta ca parametru un numar de milisecunde asa ca delay(1000) inseamna ca programul va sta pe loc timp de o secunda.

```
void rotate(int steps, float speed){
  //rotate a specific number of microsteps (8 microsteps per
//step) - (negative for reverse movement)
  //speed is any number from .01 -> 1 with 1 being fastest - Slower
//is stronger
  int dir = (steps > 0)? HIGH:LOW;
  steps = abs(steps);

  digitalWrite(DIR_PIN,dir);

  float usDelay = (1/speed) * 70;

  for(int i=0; i < steps; i++){
    digitalWrite(STEP_PIN, HIGH);
    delayMicroseconds(usDelay);
    digitalWrite(STEP_PIN, LOW);
    delayMicroseconds(usDelay);
  }
}
```

Functia *rotate* accepta ca parametrii steps de tip int si speed de tip float. Se stabileste directia de rotatie, daca steps are o valoare mai mare ca zero atunci dir va trece in "1" logic, invers daca steps este negativ, dir va trece in "0" logic. Se transmite directia catre EasyDriver prin:

```
digitalWrite(DIR_PIN,dir);
```

Daca *dir* este 1 atunci motorul se va roti intr-un sens iar daca dir trece in 0 motorul isi va schimba sensul de rotatie.

Bucla *for* transmite numarul de pasi. Observam o variabila *usDelay*. Sa presupunem ca speed = 1 (viteza foarte mare). Atunci *usDelay* ar fi egal cu 70. Dar daca *speed* = 0.5 atunci *usDelay* = 140. Am

scos in evidenta lucrul asta pentru ca o intarziere mai mare este echivalenta cu o viteza mai mica.

```
void rotateDeg(float deg, float speed){
//rotate a specific number of degrees (negitive for reverse
//movement)
//speed is any number from .01 -> 1 with 1 being fastest -   Slower
//is stronger
  int dir = (deg > 0)? HIGH:LOW;
  digitalWrite(DIR_PIN,dir);
  int steps = abs(deg)*(1/0.225);
  float usDelay = (1/speed) * 70;

   for(int i=0; i < steps; i++){
     digitalWrite(STEP_PIN, HIGH);
     delayMicroseconds(usDelay);
     digitalWrite(STEP_PIN, LOW);
     delayMicroseconds(usDelay);
  }
}
```

Functia rotateDeg este asemanatoare. Diferenta sta in parametrii de intrare si in linia de cod care transforma numarul de grade in numarul de pasi:

```
int steps = abs(deg)*(1/0.225);
```

Restul liniilor functioneaza pe acelasi principiu.

```
////////////////////////////////////////////////////////////////////
//©2011 bildr
//Released under the MIT License - Please reuse change and share
//Using the easy stepper with your arduino
//use rotate and/or rotateDeg to controll stepper motor
//speed is any number from .01 -> 1 with 1 being fastest -
//Slower Speed == Stronger movement
////////////////////////////////////////////////////////////////////

#define DIR_PIN 2
#define STEP_PIN 3

void setup() {
  pinMode(DIR_PIN, OUTPUT);
  pinMode(STEP_PIN, OUTPUT);
}

void loop(){
  //rotate a specific number of degrees
  rotateDeg(360, 1);
  delay(1000);
  rotateDeg(-360, 0.1);  //reverse
  delay(1000);
```

```
//rotate a specific number of microsteps (8 microsteps per step)
//a 200 step stepper would take 1600 micro steps for one full
//revolution

 rotate(1600, 0.5);
 delay(1000);

 rotate(-1600, 0.25); //reverse
 delay(1000);
}

void rotate(int steps, float speed){
  //rotate a specific number of microsteps (8 microsteps per
//step) - (negitive for reverse movement)
  //speed is any number from .01 -> 1 with 1 being fastest - Slower
//is stronger

  int dir = (steps > 0)? HIGH:LOW;
  steps = abs(steps);
  digitalWrite(DIR_PIN,dir);
  float usDelay = (1/speed) * 70;

  for(int i=0; i < steps; i++){
    digitalWrite(STEP_PIN, HIGH);
    delayMicroseconds(usDelay);
    digitalWrite(STEP_PIN, LOW);
    delayMicroseconds(usDelay);
  }
}
void rotateDeg(float deg, float speed){
  //rotate a specific number of degrees (negitive for reverse
//movement)
  //speed is any number from .01 -> 1 with 1 being fastest - Slower
//is stronger

  int dir = (deg > 0)? HIGH:LOW;
  digitalWrite(DIR_PIN,dir);

  int steps = abs(deg)*(1/0.225);
  float usDelay = (1/speed) * 70;

  for(int i=0; i < steps; i++){
    digitalWrite(STEP_PIN, HIGH);
    delayMicroseconds(usDelay);
    digitalWrite(STEP_PIN, LOW);
    delayMicroseconds(usDelay);
  }
}
```