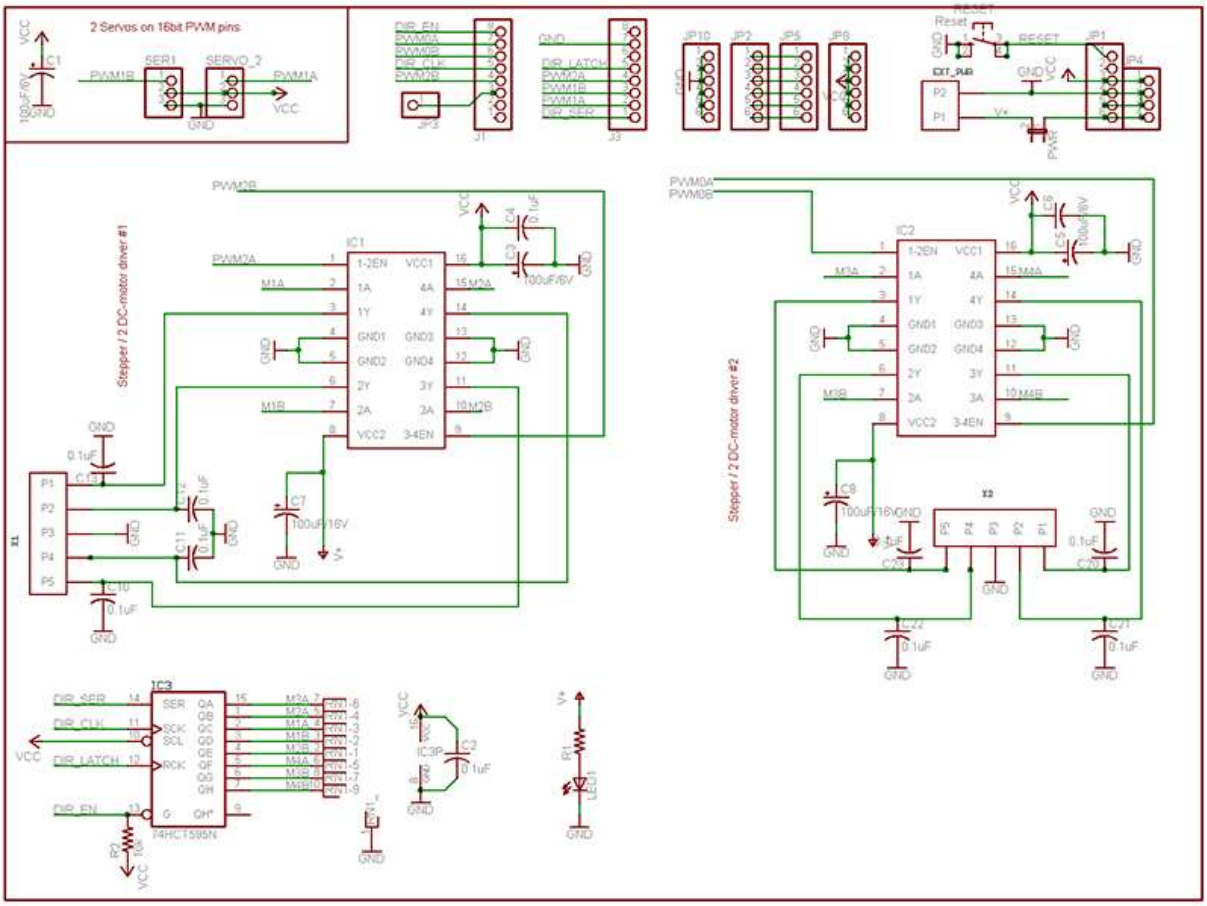
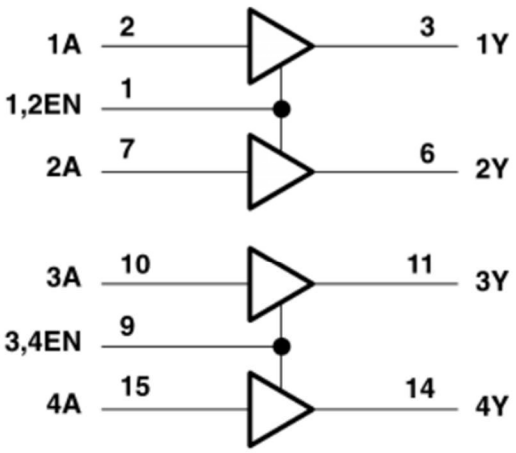
L293D motor control shield motor drive expansion board FOR uno motor shield

L293D е монолитно интегриран 4-канален драйвер, отличаващ се с високо напрежение и висок ток. Това означава, че този чип може да използвате при постоянни двигатели и захранвания до 16 волта, както и при захранване на максимален ток от 600mA. L293D чипът е също така известен като тип H -Bridge. Това е електрическа верига, която позволява на двигателите с постоянен ток да работят напред или назад. Схематичната схема е следната:





Logic diagram

FUNCTION TABLE
(each driver)

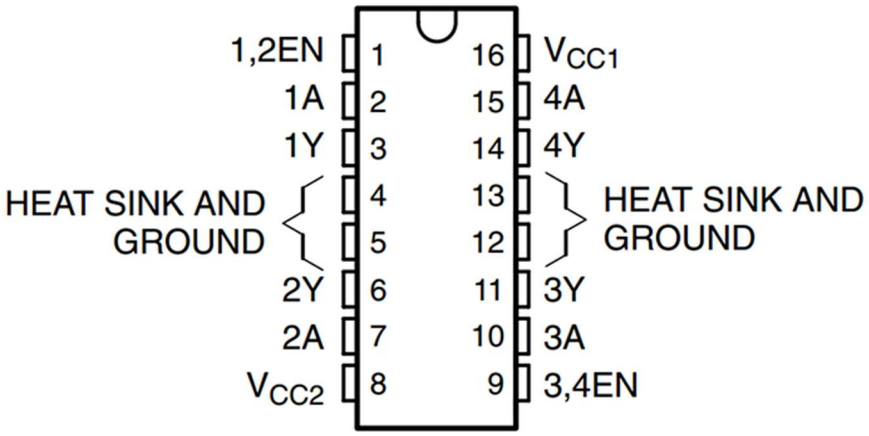
INPUTS†		OUTPUT
A	EN	Y
H	H	H
L	H	L
X	L	Z

H = high level, L = low level, X = irrelevant,
Z = high impedance (off)

† In the thermal shutdown mode, the output is in the high-impedance state, regardless of the input levels.

Function Table

Функция на пин



щифт	име	функция
1	Enable1,2	Активиране на пин за управление на 1,2 драйвера
2	Вход 1A	Вход за управление 1Y
3	Изход 1Y	Изход, свържете се с мотор
4	GND	Заземяване и радиатор
5	GND	Заземяване и радиатор
6	Изход 2Y	Изход, свържете се с мотор
7	Вход 2A	Вход за управление 2Y
8	Vcc2	Изходно захранващо напрежение
9	Enable3,4	Активиране на щифт за управление на 3,4 драйвера
10	Вход 3A	Вход за управление 3Y

11	Изход 3Y	Изход, свържете се с мотор
12	GND	Заземяване и радиатор
13	GND	Заземяване и радиатор
14	Изход 4Y	Изход, свържете се с мотор
15	Вход 4A	Вход за управление 4Y
16	Vcc1	Захранващо напрежение (7 max)

Връзки за захранване на двигателя

Напрежението на двигателя трябва да бъде между 4,5 - 25V. Тази мощност може да бъде споделена с Arduino или да бъде отделна. Ако се спрете н това, потърсете джъмпер в близост до 2 клемния захранващ конектор с надпис PWR.

Когато този джъмпер е инсталиран, захранването от жака на Arduino DC също е свързано към моторите. Вин (отбелязан с 9V на таблото) е изведен от Arduino за захранване на моторите.

Когато джъмперът е свален, това изолира мощността на двигателя от Arduino и трябва да се захранва отделно чрез свързване на захранването към 2 конектора за захранване. *Забележка: Не прилагайте захранване към 2 конектора за захранване с джъмпера на мястото си, защото това ще свърши двете захранвания заедно!*

1 x 2 клема EXT_PWR (мощност на двигателя)

- **+ M** = мотор Vcc, който трябва да бъде между 4,5 и 25V.
- **GND** = Двигател.

DC моторни връзки

Връзките на двигателя са посредством два винтови клеми за всеки двигател и са обозначени с M1 през M4. Централният терминал на 5-позиционните терминални блокове е свързан към земята.

Окабеляването на проводника на мотора и свързването с терминала е донякъде произволно и зависи от посоката на работа на двигателя. Ако моторът се върти в обратна посока от тази, която очаквате, просто обърнете окабеляването.

1 x 2 клема M1 - M4 (DC мотор 1-4)

- Мотор '-' положително олово
- Двигател '+' отрицателно олово

Връзки със стъпков двигател

Стъпковите двигатели обикновено са 4 жични. Намотка 1 ще се свърже през един моторен порт като M1 (M3), а бобината 2 ще се свърже през другия моторен отвор като M2 (M4). Ако стъпковият мотор има 5 проводника, централният проводник ще бъде свързан към централния заземен терминал.

1 x 2 клема M1 (M3) - M2 (M4) (серво мотор 1-2)

- Моторна намотка 1
- Моторна намотка 2

Arduino към shield пин връзки

Shield има бутон за дистанционно нулиране, разположен върху него за лесен достъп.

Той използва пинове D3, D4, D5, D6, D7, D8, D11 и D12 за управление на постоянен и постоянен двигател.

D9 и D10 излизат в заглавките на серво.

Останалите щифтове са налични, включително 6-те аналогови щифта, които могат да се използват и като цифрови I / O. Те имат подложки за запояване, така че при желание за лесно свързване може да се добави заглавка. До него е ред от 5V и Ground връзки, които също могат да бъдат запълнени с заглавки, ако желаете.

Примерна програма за защита на драйвера на двигателя L293D V1

```
/*
Exercise V1 L293D Motor Shield
This simply creates 4 motor objects and then runs them forward, backwards,
stops them and then repeats.
*/
#include <AFMotor.h>


const int MOTOR_1 = 1;
const int MOTOR_2 = 2;
const int MOTOR_3 = 3;
```

```
const int MOTOR_4 = 4;

AF_DCMotor motor1(MOTOR_1, MOTOR12_64KHZ); // create motor object, 64KHz pwm
AF_DCMotor motor2(MOTOR_2, MOTOR12_64KHZ); // create motor object, 64KHz pwm
AF_DCMotor motor3(MOTOR_3, MOTOR12_64KHZ); // create motor object, 64KHz pwm
AF_DCMotor motor4(MOTOR_4, MOTOR12_64KHZ); // create motor object, 64KHz pwm
//=============================================================================
//  Initialization
//=============================================================================
void setup() {
  Serial.begin(9600);          // Initialize serial port
  Serial.println("Motor test");

  motor1.setSpeed(200);         // set the motor speed to 0-255
  motor2.setSpeed(200);
  motor3.setSpeed(200);
  motor4.setSpeed(200);
}
//=============================================================================
//  Main
//=============================================================================
void loop() {
  // Simply run the selected motor in both directions and stop.  Then repeat
  Serial.println("Forward");

  motor1.run(FORWARD);        // turn it on going forward
  motor2.run(FORWARD);
  motor3.run(FORWARD);
  motor4.run(FORWARD);
  delay(3000);
```

```
  Serial.println("Reverse");
  motor1.run(BACKWARD);      // the other way
  motor2.run(BACKWARD);
  motor3.run(BACKWARD);
  motor4.run(BACKWARD);
  delay(3000);

  Serial.println("Stop");
  motor1.run(RELEASE);       // stopped
  motor2.run(RELEASE);
  motor3.run(RELEASE);
  motor4.run(RELEASE);
  delay(3000);
}
```