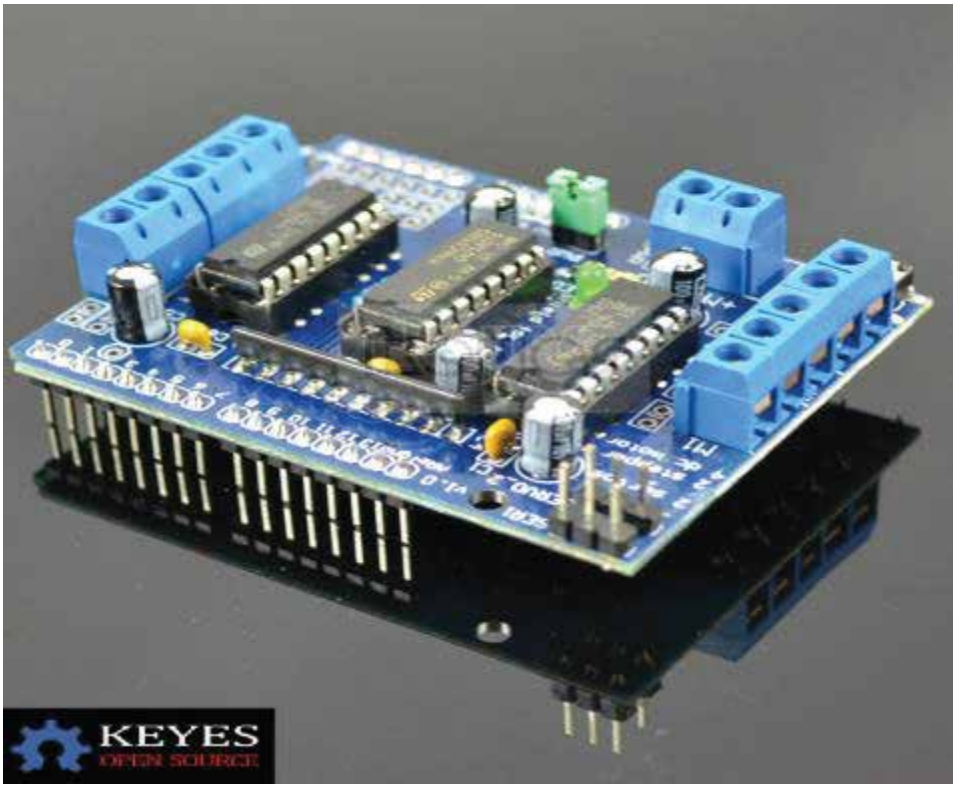
L203D Motor Control Shield



Introduction:

Arduino is a good introduction to electronic production. With this shield, it can be a good platform for robot development. Herein, we demonstrate a fully functional motor control shield that can drive a variety of simple and even slightly complex projects.

This commonly-used DC motor drive module adopts 293D chip, which is a small current DC motor drive chip. Its pins are Arduino-compatible, so it is easy for enthusiasts to quickly develop Arduino -based products.

Features:

* Two 5V servo ports connected to the Arduino high-resolution & high-precision timer - no jitter!

* Up to 4 bi-directional DC motors and 4-way PWM speed controls (resolution: approx 0.5%)

* Two stepper motor control rotation direction, single/double step, staggered/micro-step and rotation Angle.

* 4-way H-bridge: The L293D chip provides 0.6A (peak 1.2A) current per bridge with thermal outage protection, 7V to 12V.

* The pull-down resistor ensures that the motor stays stopped when powered on.

* Large terminal interfaces simplify wiring (10-22 AWG) and power supply.

* Arduino reset button is reserved.

* 2 large terminal external power ports ensure the separation of logic and motor drive power.

* Compatible with Mega, Diecimila, and Duemilanove.

* Download easy-to-use Arduino library for project development

Application: Arduino beginners, Arduino experimental equipment platform, Arduino interactive electronics, Arduino robot, etc.

Features: multiple functions, simple operations, a powerful driver library support and function updating.

The L293D is able to drive a 4-way DC motor or a 2-way stepper motor while driving a 2-way servo. It supports the latest Arduino UNO and Arduino Mega 2560.

Possible Collocations of Driving:

* 4-way DC motor and 2-way servo

* 2-way DC motor, 2-way stepper motor and 2-way servo

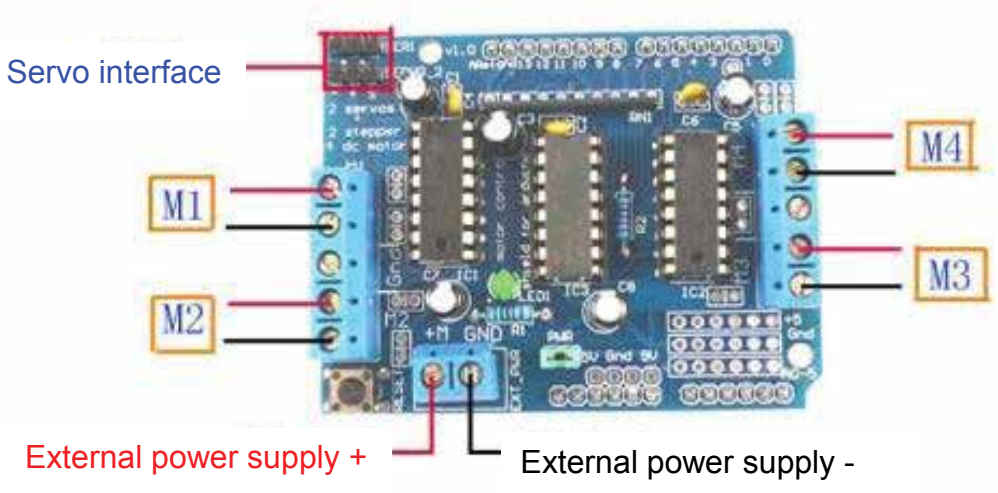
* 2-way stepper motor and 2-way servo

Here is an experiment of driving four DC motors with L293D.

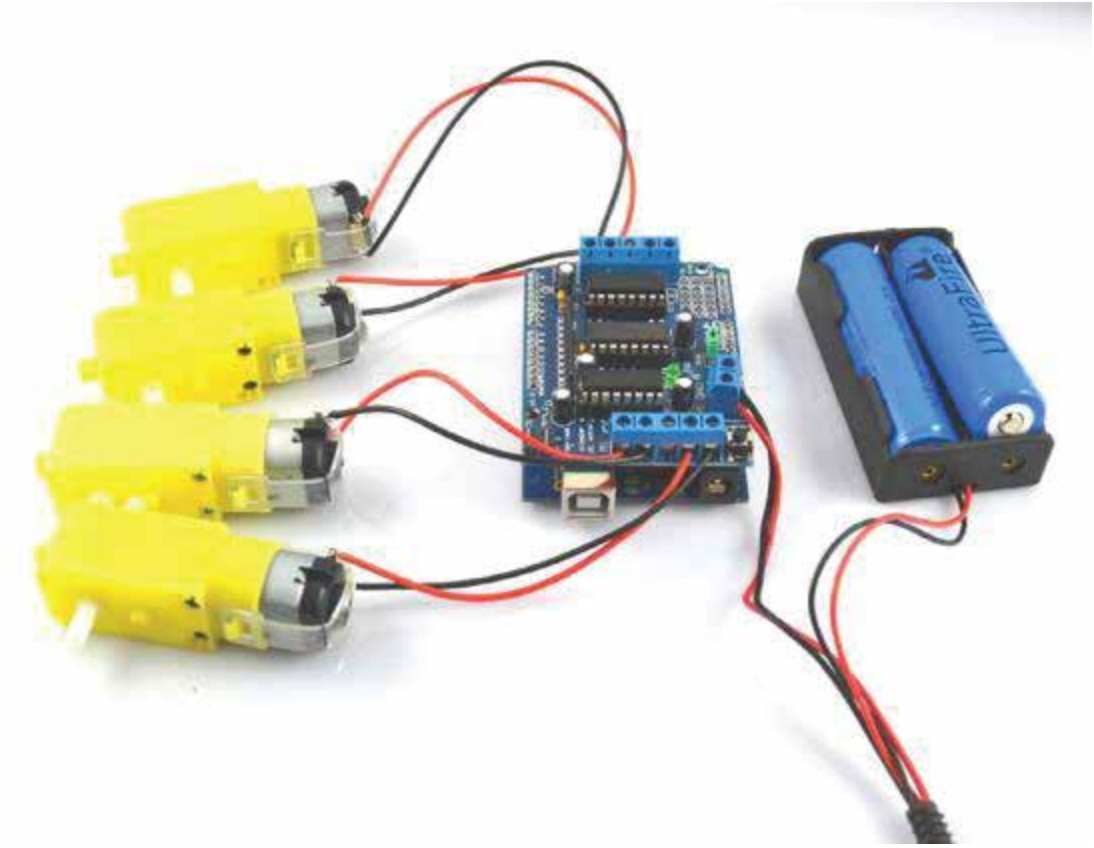
Required Components:

Arduino controller * 1	L293D module * 1
DC motor * 4	9V power supply * 1

Then wiring up accordingly:



Wiring Diagram:



Test Code:

```
#include <Servo.h>
#define MOTORLATCH  12
#define MOTORCLK 4
#define MOTORENABLE 7
#define MOTORDATA 8
#define MOTOR1_A 2
#define MOTOR1_B 3
#define MOTOR2_A  1
#define MOTOR2_B 4
#define MOTOR3_A 5
#define MOTOR3_B 7
#define MOTOR4_A 0
#define MOTOR4_B 6
#define MOTOR1_PWM  11
#define MOTOR2_PWM 3
```

```
#define MOTOR3_PWM 6
#define MOTOR4_PWM 5
#define SERVO1_PWM  10
#define SERVO2_PWM 9
#define FORWARD  1
#define BACKWARD 2
#define BRAKE 3
#define RELEASE 4
Servo servo_1;
Servo servo_2;
void setup()
{
   Serial.begin(9600);
   Serial.println("Simple Adafruit Motor Shield sketch");
   servo_ 1.attach(SERVO1_PWM);
   servo_2.attach(SERVO2_PWM);
}
void loop()
{
   motor(1, FORWARD, 255);
   motor(2, FORWARD, 255);
   motor(3, FORWARD, 255);
   motor(4, FORWARD, 255);
   delay(2000);
   // Be friendly to the motor: stop it before reverse.
   motor(1, RELEASE, 0);
   motor(2, RELEASE, 0);
   motor(3, RELEASE, 0);
   motor(4, RELEASE, 0);
   delay(100);
   motor(1, BACKWARD, 128);
   motor(2, BACKWARD, 128);
   motor(3, BACKWARD, 128);
   motor(4, BACKWARD, 128);
   delay(2000);
   motor(1, RELEASE, 0);
   motor(2, RELEASE, 0);
   motor(3, RELEASE, 0);
   motor(4, RELEASE, 0);
   delay(100);
}
void motor(int nMotor, int command, int speed)
{
   int motorA, motorB;
```

```
if (nMotor >= 1 && nMotor <= 4)
 {
   switch (nMotor)
    {
   case 1:
      motorA     = MOTOR1_A;
      motorB     = MOTOR1_B;
      break;
   case 2:
      motorA     = MOTOR2_A;
      motorB     = MOTOR2_B;
      break;
   case 3:
      motorA     = MOTOR3_A;
      motorB     = MOTOR3_B;
      break;
   case 4:
      motorA     = MOTOR4_A;
      motorB     = MOTOR4_B;
      break;
   default:
      break;
   }
   switch (command)
    {
   case FORWARD:
      motor_output (motorA, HIGH, speed);
      motor_output (motorB, LOW, -1);          // -1: no PWM set
      break;
   case BACKWARD:
      motor_output (motorA, LOW, speed);
      motor_output (motorB, HIGH, -1);        // -1: no PWM set
      break;
   case BRAKE:

      motor_output (motorA, LOW, 255); // 255: fully on.
      motor_output (motorB, LOW, -1);    // -1: no PWM set
      break;
   case RELEASE:
      motor_output (motorA, LOW, 0);    // 0: output floating.
      motor_output (motorB, LOW, -1); // -1: no PWM set
      break;
   default:
```

```
            break;
        }
    }
}
void motor_output (int output, int high_low, int speed)
{
    int motorPWM;

    switch (output)
    {
    case MOTOR1_A:
    case MOTOR1_B:
        motorPWM = MOTOR1_PWM;
        break;
    case MOTOR2_A:
    case MOTOR2_B:
        motorPWM = MOTOR2_PWM;
        break;
    case MOTOR3_A:
    case MOTOR3_B:
        motorPWM = MOTOR3_PWM;
        break;
    case MOTOR4_A:
    case MOTOR4_B:
        motorPWM = MOTOR4_PWM;
        break;
    default:
        speed = -3333;
        break;
    }

    if (speed != -3333)
    {

        shiftWrite(output, high_low);

        // set PWM only if it is valid
        if (speed >= 0 && speed <= 255)
        {
            analogWrite(motorPWM, speed);
        }
    }
}
void shiftWrite(int output, int high_low)
```

```
{
  static int latch_copy;
  static int shift_register_initialized = false;

  // Do the initialization on the fly,
  // at the first time it is used.
  if (!shift_register_initialized)
  {
    // Set pins for shift register to output
    pinMode(MOTORLATCH, OUTPUT);
    pinMode(MOTORENABLE, OUTPUT);
    pinMode(MOTORDATA, OUTPUT);
    pinMode(MOTORCLK, OUTPUT);

    // Set pins for shift register to default value (low);
    digitalWrite(MOTORDATA, LOW);
    digitalWrite(MOTORLATCH, LOW);
    digitalWrite(MOTORCLK, LOW);
    // Enable the shift register, set Enable pin Low.
    digitalWrite(MOTORENABLE, LOW);

    // start with all outputs (of the shift register) low
    latch_copy = 0;

    shift_register_initialized = true;
  }

  // The defines HIGH and LOW are 1 and 0.
  // So this is valid.
  bitWrite(latch_copy, output, high_low);
  shiftOut(MOTORDATA, MOTORCLK, MSBFIRST, latch_copy);
  delayMicroseconds(5);      // For safety, not really needed.
  digitalWrite(MOTORLATCH, HIGH);
  delayMicroseconds(5);      // For safety, not really needed.
  digitalWrite(MOTORLATCH, LOW);
}
```

Distributed by:
Electus Distribution Pty Ltd
46 Eastern Creek Dr,
Eastern Creek NSW 2766 Australia
1300 738 555 I electusdistribution.com.au

Made in China