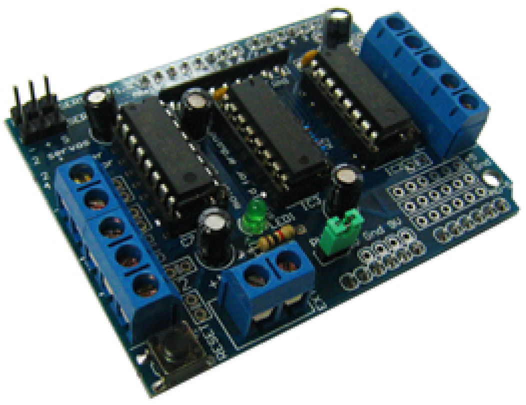
L293D 馬達驅動控制擴展板



介紹：

Arduino 是一款很好的電子製作入門，有了馬達驅動擴展板可以善用為很好的機器人開發平台。這裡介紹一款能驅動各種簡單到稍複雜項目的全功能的馬達驅動擴展板。
這是一款常用的直流馬達驅動模組，採用 L293D 小電流直流馬達驅動芯片。連接腳已被做成可與 Arduino 兼容以方便快速開發。

特點：

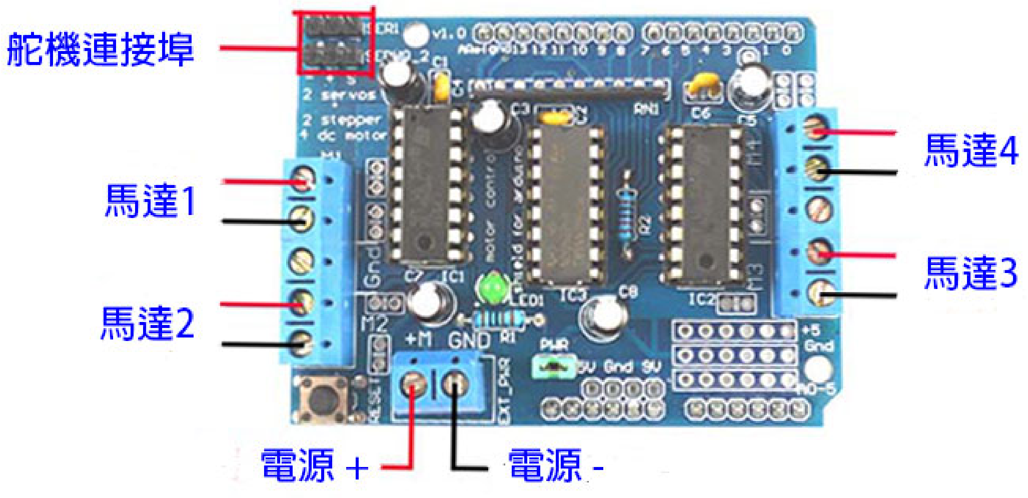
1、二個直流 5V 伺服馬達(舵機)接埠連接到 Arduino 的高解析高精度的定時器-無抖動！
2、具有四個雙向直流馬達及 4 路 PWM 調速（大約 0.5%的解析度）
3、具有二個步進馬達正反轉控制，單/雙步控制，交錯或微步及旋轉角度控制。
4、四路 H-橋：L293D 芯片每路橋提供 0.6A（峰值 1.2A）電流並且帶有熱斷電保護，4.5V to 36V。
5、下拉電阻保證在上電時馬達保持停止狀態。
6、大終端接線端子使接線更容易（10 - 22AWG）和電源。
7、帶有 Arduino 復位按鈕。
8、二個大終端外部電源接線端子保證邏輯和馬達驅動電源分離。
9、兼容 Mega, Diecimila, & Duemilanove。
10、下載方便使用的 Arduino 軟體庫快速進行項目開發

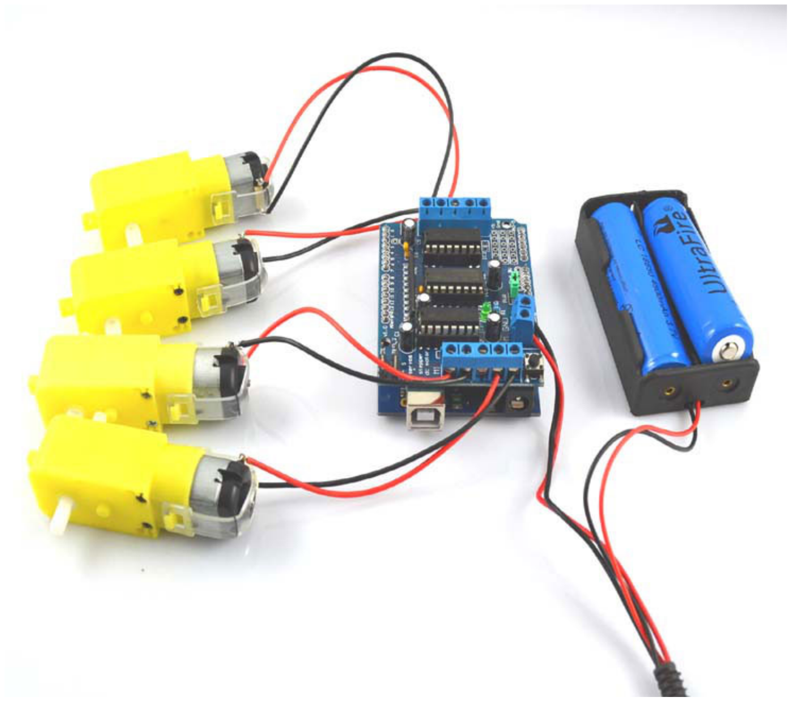
適用範圍： Arduino 初學者，Arduino 實驗器材平台，Arduino 互動電子，Arduino 機器人等。

特點：功能多，操作方便，有強大的驅動庫支持及功能更新。
L293D 可驅動四路直流馬達或者二路步進馬達，同時還能驅動二路舵機。
支持最新 Arduino UNO, Arduino Mega 2560

您可以這樣搭配使用：
驅動四路直流馬達和兩路舵機
驅動兩路直流馬達和一路步進馬達和兩路舵機
驅動兩路步進馬達和兩路舵機

以下是利用 L293D 驅動四個直流馬達實驗
準備清單：
Arduino 控制器一個
L293D 馬達驅動模組一個
直流馬達四個
直流 9V 電源一個




實際接線圖

程序代碼如下：

```
#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);
}
```