AA031

L293D Motor Driver Board

V1.0.23.10.20

Preface

OurCompany

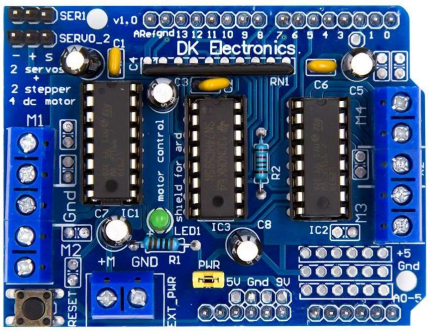
KUONGSHUN Electronic Company is a supplier and manufacturer of electronic components, it is committed to board and starter kit for Arduino, Raspberry PI, Smart Robot Car, 3D printer. It is also a collection of scientific research, design, production,

maintenance and sales of high-tech enterprises, in the field of automation with professional standards and mature technology, we rapid rise in the field of foreign trade. Relying on technology and development, continuing to provide users with high-tech products, is our constant pursuit. Fully introduction of foreign advanced technology to enhance the value of our products.

Company gains users' praise for supplying first-class quality product and superb technical services, has now become the first choice of domestic and international procurement company.

Official Website: https://www.kuongshun-ks.com

AA031 L293D Motor Driver Board



Product Description

This is a general-purpose motor driver board that can drive 4 DC motors, or drive 2 stepper motors, and can also drive 2 servo modules. In order to facilitate the hobbyist's arduino-based development, its pins are made compatible with arduino.

When using it, just stack it on arduino uno or arduino mega 2560, easy to operate. In order to facilitate the control, we provide powerful driver library support, it is suitable for arduino beginners, arduino experimental equipment platform, arduino interactive electronics, arduino robot and so on.

Product Parameters

2 x 5V servo ports

4 bi-directional DC motors each with 8-bit speed selection

2 stepper motors (unipolar or bipolar) single coil, dual coil, staggered or microstepping

4 H-Bridges: L293D chip provides .0.6A (1.2A peak) per bridge with thermal cutoff protection, 4.5V to 36V

Size: 70mm*53mm

Pull-down resistor ensures that the motor stays stopped during power-up.

Arduino reset button

2 external power supply terminals ensure separation of logic and motor drive power supplies

Compatible with UNO, Mega, Diecimila, Duemilanove

Easy-to-use Arduino software library available for download

Tip before use

To use this driver, you need to install the library files into Arduino IDE. Just extract the library files to the libraries folder in the Arduino installation directory, e.g. C:\ProgramFiles\arduino\libraries, and then reopen the Arduino IDE to use it.

Voltage requirements: motors require a lot of energy, especially cheap motors because they are very inefficient. The first thing you need to do is figure out what voltage your motor will be using.

Current Requirements: Next you need to figure out how much current your motors are going to draw. The driver chips that come with the driver boards are designed to provide up to 600mA per motor and 1.2A peak current. When you get close to 1A, you should add a heat sink to the driver chip. Otherwise it will fail thermally and burn the driver chip.

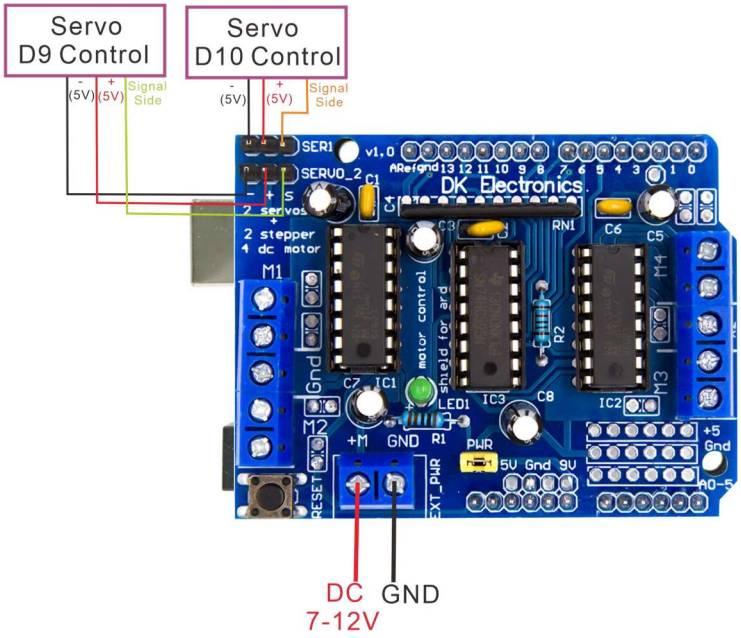
How do I set up an Arduino + Shield driver board to power the motors? The servos are powered by the regular 5V used by the Arduino. It doesn't matter for small hobby servos. If you are using a more powerful one, you should connect a separate power supply! DC motors are powered by a high power supply, not the regular 5V supply. Motors require higher currents, and a power supply that is too low in power may burn out the power supply, or not drive the motor. There are two ways to power your motors with a high power supply. One is the DC socket on the Arduino board the other is the 2-wire terminal on the motor driver board labeled EXT_PWR. The DC socket on the Arduino board has a protection diode, so it won't mess things up if you connect the wrong power supply. Be careful though, the EXT_PWR terminal on the driver board does not have a protection diode, so be very careful not to connect it backwards or you could potentially burn your driver board or Arduino!

If you just want to use a separate DC power supply to power the Arduino and motors, simply plug the power supply into the DC socket on the Arduino or the 2-wire terminal EXT_PWR on the driver board. Then plug in the power jumper on the driver board. If you are using a Diecimila Arduino, set the Arduino power jumper to EXT. note that you may experience an Arduino reset if the power supply is low, so this method of powering the Arduino is not recommended. If you want the Arduino to be powered by USB, the motor is powered by another DC power supply. Plug in the USB adapter cable and connect the power supply to the PWR_EXT terminal on the driver board, being careful not to put the jumpers on the driver board at this point. This is the recommended method to power your motor project. (If you are using Diecimila Arduino, don't forget to set the Arduino power jumper to USB, if you are using Diecimila Arduino you can plug the DC power supply to the Arduino and then put the power jumper on the driver board). If you want to use two separate DC power supplies to power the Arduino and the motor, plug the power supply into the DC socket on the Arduino and connect the motor power supply to the PWR_EXT terminal on the driver board, making sure to remove the jumper on the motor driver board. If you are using a Diecimila Arduino, set the Arduino jump to EXT. this is the recommended way to power your motor project. Either way is fine, if you want to use a DC motor/stepper motor system, the LED on the motor driver board should be glowing to indicate that the motor is powered properly.

Driving Servo

There are two 3-wire interfaces on the driver board for driving the servo, corresponding to GND, 5V, and signal terminals respectively. These two signal terminals correspond to D9 and D10 on the Arduino control board.

When driving the servo, the wiring is as shown below.



Special note: At this point, just the EXT_PWR interface can be used to supply power, but also the black DC header of the Arduino control board can be used to input 7-12V power.

When driving the servo motion, the test code is as follows.

```
/*
https://www.kuongshun-ks.com/
*/
#include <Servo.h>
Servo myservo;// define servo variable name void
setup()
{
myservo.attach(10);// select servo pin(9 or 10)
}
void loop()
{
myservo.write(0);// set rotate angle of the motor
delay(1000); myservo.write(180);// set rotate
angle of the motor delay(1000);
}
```

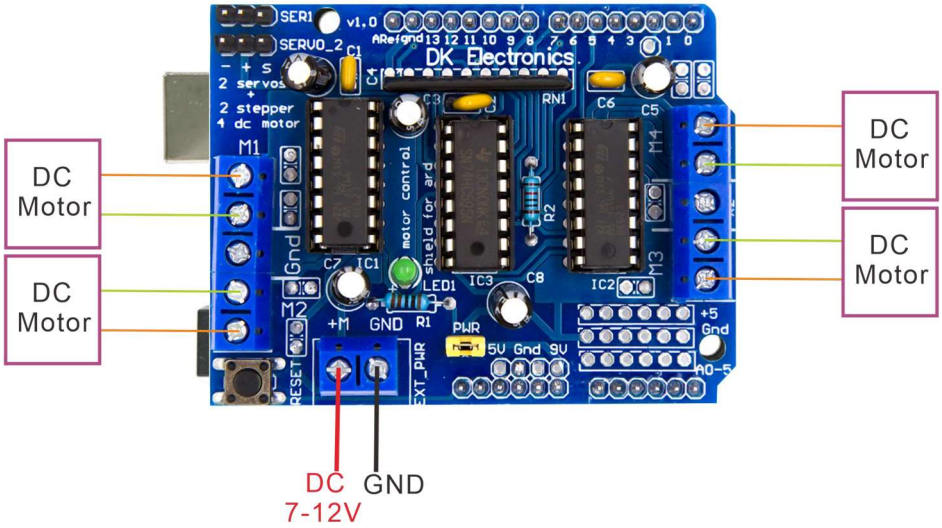
At this point in the code, it is turning the D10 console servo back and forth from 0-180° . To control the D9 console servo, simply change myservo.attach(10) to myservo.attach(9) in the code.

Driving DC motors

This driver board can drive up to 4 motors in both directions. Bi-directional means they can be turned forward and reverse. Using high quality built-in PWM, the speed can be varied in 0.5% increments, meaning that the speed is very smooth and does not change!

Note: The H-bridge chip is not really loadable over 0.6A or peak over 1.2A, so it is more suitable for small power motors.

When driving a DC motor movement, the wiring is as shown below.



Special note: At this point, just the EXT_PWR interface can be used to supply power, but also the black DC header of the Arduino control board can be used to input 7-12V power.

When driving a DC motor, the test code is as follows.

```
/*
https://www.kuongshun-ks.com/
*/
#include <AFMotor.h>
AF_DCMotor motor(4);
void setup() {
  Serial.begin(9600);           // set up Serial library at 9600 bps
  Serial.println("Motor test!");
  // turn on motor
  motor.setSpeed(200);
  motor.run(RELEASE);
}

void loop() {
  uint8_t i;
  Serial.print("tick");
  motor.run(FORWARD);
  for (i=0; i<255; i++) {
    motor.setSpeed(i); delay(10);
 }
  for (i=255; i!=0; i--) {
    motor.setSpeed(i); delay(10);
 }
  Serial.print("tock");
  motor.run(BACKWARD);
```

```
  for (i=0; i<255; i++) {
     motor.setSpeed(i); delay(10);
  }
  for (i=255; i!=0; i--) {
     motor.setSpeed(i); delay(10);
  }
  Serial.print("tech");
  motor.run(RELEASE); delay(1000);
}
```

The code setup is described below.

Make sure to #include <AFMotor.h> header file.

Use AF_DCMotor motor(4) to create a DC Motor object. 4 means to control the motor of the M4 interface, similarly if you need to control the motors of M1 M2 and M3, just replace 4 with 1 2 3. Then you can motor.setSpeed(speed) function to set the speed of the motor, speed range from 0 (stop) to 255 (full speed), you can set the speed you want.

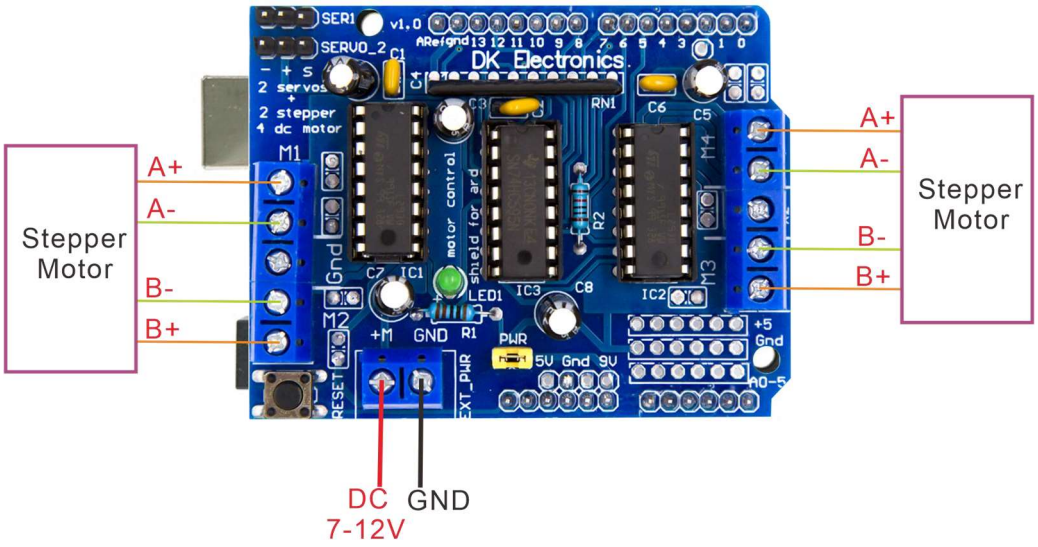
The motor is turned on by calling motor.run(direction), the direction is FORWARD, BACKWARD or RELEASE. of course the Arduino doesn't actually know if the motor is spinning forward or reverse. So if you want to change the direction you think is forward, just swap the motor leads.

Driving stepper motors

Stepper motors are excellent for high precision control. Also ideal for many robotics and CNC projects. This driver board supports up to 2-way stepper motors, and the library can support both two-phase and unipolar stepper motors very well. To hook up a stepper motor, first figure out which windings (coils) those pins are connected to and which pins are connected to the intermediate windings (coils). If it is a 5 wire stepper motor then there will be 1 center tap for both sets of coils. There are many tutorials on how to make connections to the coil pins. The center tap should be connected to the output GND terminal of the motor driver board and then coil 1 should be connected to one of the motor ports (M1 or M3) coil 2 should be connected to the other motor port (M2 or M4).

Bipolar stepper motors are the same as unipolar stepper motors except that they do not have a fifth lead connected to ground, the code is the same.

To drive the stepper motor motion, the wiring is as shown below.



Special note: At this point, just the EXT_PWR interface can be used to supply power, but also the black DC header of the Arduino control board can be used to input 7-12V power.

When driving a stepper motor, the test code is as follows.

```
/*
https://www.kuongshun-ks.com/
*/
#include <AFMotor.h>
// Connect a stepper motor with 48 steps per revolution (7.5 degree)
// to motor port #2 (M3 and M4)
AF_Stepper motor(48, 2); void
setup() {
  Serial.begin(9600);              // set up Serial library at 9600 bps
  Serial.println("Stepper test!");
  motor.setSpeed(10);          //      10      rpm
  motor.step(100, FORWARD, SINGLE);
  motor.release(); delay(1000);
}

void loop() {
  Serial.println("Single coil steps");
  motor.step(100, FORWARD, SINGLE); motor.step(100,
  BACKWARD, SINGLE);
  Serial.println("Double coil steps");
  motor.step(100, FORWARD, DOUBLE); motor.step(100,
  BACKWARD, DOUBLE);
  Serial.println("Interleave coil steps");
  motor.step(100, FORWARD, INTERLEAVE); motor.step(100,
  BACKWARD, INTERLEAVE);
  Serial.println("Micrsostep steps");
  motor.step(100, FORWARD, MICROSTEP);
```

```
    motor.step(100, BACKWARD, MICROSTEP);
  }
```

The code setup is described below.

Make sure to #include <AFMotor.h> header file

Use AF_Stepper motor(steps, stepper#) to create a stepper motor object, the constructor takes two parameters. The first one is how many steps per revolution the stepper motor needs, a 7.5 degree/step motor has 360/7.5=48 steps. stepper# is the port to connect to. If you use M1 and M2 he is port 1, if you use M3 and M4 he is port 2.

Set the number of revolutions per minute you want the stepper motor to rotate by using the motor.setSpeed(rpm) function.

Then each time you want the stepper motor to rotate just call the motor.step(#steps, direction, steptype) function. #steps is the number of steps you want the stepper motor to take, direction is FORWARD or BACKWARD, and steptype is SINGLE, DOUBLE, INTERLEAVE, or MICROSTEP. "Single" means single coil. Single" means single coil enabled, "double" means double coil enabled (for high torque), and "interleave" means alternating between single and double to get double the accuracy (but of course halving the speed). "Microstep" is a way to use PWM between each step to create smooth motion. By default the motor will stay in the position it was in when it stopped. If you want to release all the coils then you need to call the motor.release() function. The step command will return as soon as the movement is complete.

Resources

Related code and library files are linked below:

Xxxxxxxxxxxx