POLOLU DRV8835 DUAL MOTOR DRIVER SHIELD FOR ARDUINO

USER'S GUIDE

USING THE SHIELD

The shield plugs into Arduino digital pins 6, 7, 8, 9, and 10 on one side and Arduino VIN, GND, GND, and 5V/VCC on the other. The upper-left corner of the shield partially blocks the Arduino's 3.3V pin, but this region of the board (marked with a white silkscreen box) can be removed if necessary to allow access. The shield also blocks Arduino digital pin 6, but it provides alternate access points to this pin via the neighboring through-holes. The board does not use pin 6 for anything.

In the shield's default state, the motor driver shield and Arduino are powered separately, though they share a common ground and the Arduino's 5V rail serves as the shield's logic supply. When used this way, the Arduino must be powered via USB, its power jack, or its VIN pin, and the shield must be supplied with 1.5 V to 11 V through its large VIN and GND pads. Attempting to power the shield from the Arduino is not recommended as this could result in large currents flowing through small traces. However, if the motor power supply is suitable, it is possible to power the Arduino from the shield. This can be accomplished by placing a jumper between the shield pins in the lower-left corner labeled VOUT and AVIN, which connects the reverse-protected motor supply voltage to the Arduino's VIN pin to power the Arduino. The Arduino's power jack must remain disconnected at all times in this configuration.

Warning: When powering the Arduino from the motor shield, you must **never** connect a different power supply to the Arduino's VIN pin or plug a power supply into the Arduino's power jack, as doing so will create a short between the shield's power supply and the Arduino's power supply that could permanently damage both the Arduino and the motor shield. In this case, it is also

important that your shield power supply is an acceptable voltage for your Arduino, so the full shield operating voltage range of 1.5 V to 11 V probably will not be available. For example, the recommended operating voltage of the Arduino Uno is 7 – 12 V.

By default, the board operates in PHASE/ENABLE mode, in which a PWM signal applied to the ENABLE pin determines motor speed and the digital state of the PHASE pin determines direction of motor rotation. Arduino pins 9 and 7 are used to control the speed and direction, respectively, of motor 1, and pins 10 and 8 control the speed and direction of motor 2. The table below shows how the inputs affect the outputs in this mode:

Drive/brake operation in default PHASE/ENABLE mode				
xPHASE	xENABLE	MxA	MxB	operating mode
0	PWM	PWM	L	forward/brake at speed *PWM %*
1	PWM	L	PWM	reverse/brake at speed *PWM %*
X	0	L	L	brake low (outputs shorted to ground)

PHASE/ENABLE mode should be suitable for most applications.

CONFIGURING THE BOARD FOR IN/IN MODE

The operating mode of the driver is controlled by the MODE pin, which the shield connects to VCC by default to select PHASE/ENABLE mode. To change the mode, locate the pair of 0.1″ through-holes in the upper-left part of the board labeled “MODE” and use a knife to cut the trace that connects the two on the bottom side of the PCB. Since the MODE pin has an internal pull-down resistor, severing its connection to VCC is all it takes to switch the control interface to IN/IN, which allows for slightly more advanced control options as described in the table below:

Drive/coast or drive/brake operation with MODE=0 (IN/IN)				
xIN1	xIN2	MxA	MxB	operating mode
0	0	OPEN	OPEN	coast (outputs off)
PWM	0	PWM	L	forward/coast at speed *PWM %*
0	PWM	L	PWM	reverse/coast at speed *PWM %*
1	PWM	PWM	L	forward/brake at speed *100% – PWM %*
PWM	1	L	PWM	reverse/brake at speed *100% – PWM %*
1	1	L	L	brake low (outputs shorted to ground)

Once the trace between the two pins has been cut, you can use a pair of header pins and a shorting block to control the mode: with the shorting block on, the mode is PHASE/ENABLE; with it off, the mode is IN/IN.

IN/IN mode is generally only useful if you only care about on/off control of the motors or if you can supply PWM signals to all four inputs, which is **not** possible when using the default pins on an Arduino Uno. If you want to be able to control the speed of the motors when using this mode, you should either remap the control pins or select an Arduino that can generate PWM signals with digital pins 7, 8, 9, and 10 (like the Arduino Mega 2560).

CONFIGURING THE BOARD FOR SINGLE-CHANNEL MODE (PARALLEL OUTPUTS)

In order to use the two motor channels in parallel to control a single motor, it is important to ensure that both channels will always receive the same control signals, so the reconfiguration process begins with a modification to the control inputs. First, locate the 2×5 grouping of 0.1″ through-holes along the right side of the board. These holes run parallel to pins 6-10 and the traces between them on the underside of the PCB effectively link the Arduino pins to the DRV8835 control pins. If you want to remap one of these control pins, you can cut the desired trace with a knife and then run a wire from the inner hole to a new Arduino pin. The remapping for single-channel mode requires you cut **one** PWM (9 or 10) and **one** DIR (7 or 8) trace; cut 10 and 8 to control both outputs from the motor 1 input pins, or cut 9 and 7 to control both from the motor 2 input pins. If you then solder a row of header pins along the interior row of holes, you can safely connect both PWM lines together and both DIR lines together using shorting blocks. In this configuration, the two uncut Arduino control lines determine the behavior of both motor channels.

The last step is to connect the output channels together. An easy way to do this is to solder header pins to the two pairs of holes labeled “A” and “B” near the motor outputs. Placing shorting blocks across these pairs of pins connects M1A to M2A and M1B to M2B, which in turn means you can get up to 3 A from the connection points for either channel (e.g. you can have your motor connected just to the M1A and M1B terminal blocks rather than trying to find a way to connect it to all four motor outputs).

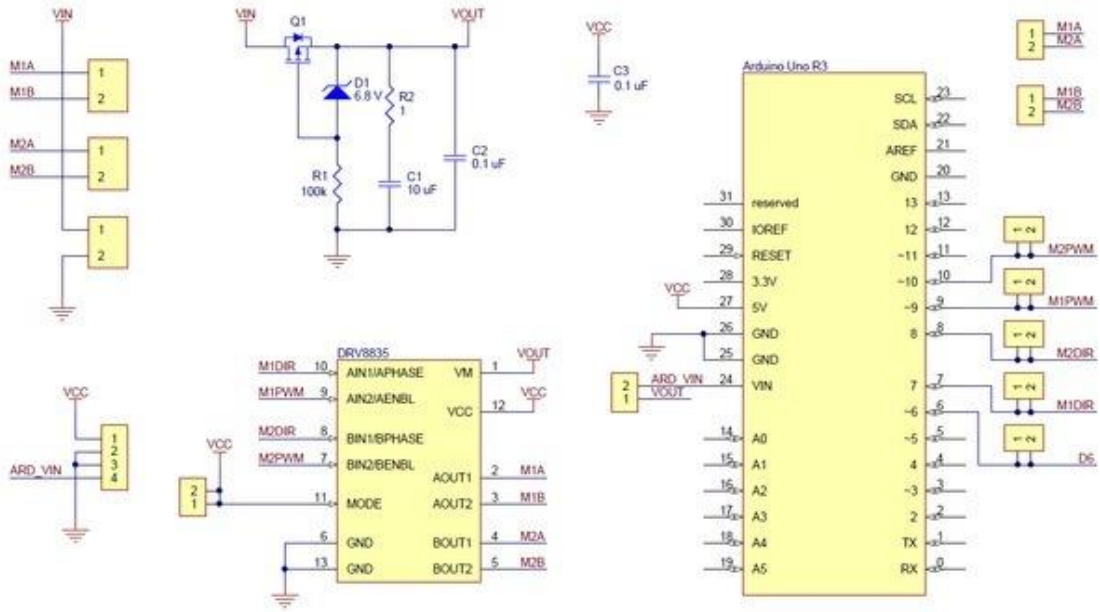
REAL-WORLD POWER DISSIPATION CONSIDERATIONS

The DRV8835 datasheet recommends a maximum continuous current of 1.5 A per motor channel. However, the chip by itself will overheat at lower currents. For example, in our tests at room temperature with no forced air flow, the chip was able to deliver 1.5 A per channel for approximately 15 seconds before the chip’s thermal protection

kicked in and disabled the motor outputs, while a continuous current of 1.2 A per channel was sustainable for many minutes without triggering a thermal shutdown. The actual current you can deliver will depend on how well you can keep the motor driver cool. The carrier's printed circuit board is designed to draw heat out of the motor driver chip, but performance can be improved by adding a heat sink. Our tests were conducted at 100% duty cycle; PWMing the motor will introduce additional heating proportional to the frequency.

> This product can get hot enough to burn you long before the chip overheats. Take care when handling this product and other components connected to it.

SCHEMATIC DIAGRAM



Pololu DRV8835 Dual Motor Driver Shield for Arduino schematic diagram.