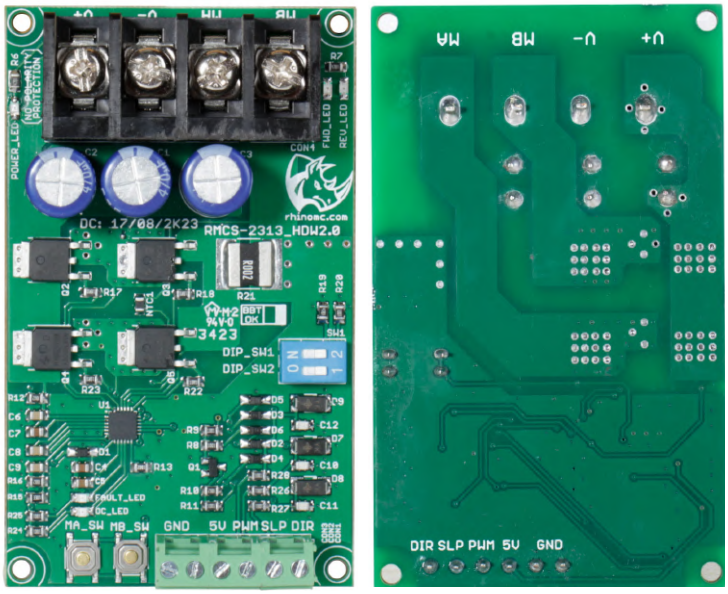
Single Channel DC Motor Driver

6V-30V 20Amp



RMCS – 2313

Operating Manual v1.0

Contents

Introduction :

The Rhino single channel Driver 20A enables control of a high-power brushed DC motor from 6V to 30V. With full discrete NMOS H-Bridge design, this motor driver is able to support 20Amp continuously for the motor without any heat-sink

The onboard test buttons and motors output LEDs allow functional test of the motor driver in a quick and convenient way without hooking up the host controller. This motor driver can be controlled with PWM and DIR inputs.

It's compatible with wide variety of host controller (e.g. Arduino, Raspberry Pi, PLC).

Various protection features are also incorporated in each channel of the Rhino 20A. Over-current protection prevents the motor driver from damage when the motor stalls or an over-sized motor is hooked up. When the motor is trying to draw current more than what the motor driver can support, the motor current will be limited at the maximum threshold.

Assisted by temperature protection, the maximum current limiting threshold is determined by the board temperature. The higher the board temperature, the lower the current limiting threshold. This way, Rhino Dual Driver 20A able to deliver its full potential depending on the actual condition without damaging the MOSFETs.

Note: Power input does not have reverse-voltage protection. Connecting the battery in reverse polarity will damage the motor driver instantaneously.

Technical Specifications :

Supply Voltage and Current :

Specification	Min	Max	Units	Comments
Supply Voltage	6	30	Volts DC	Between +VCC and GND
Phase Current (continuous)	-	20	Amps	Peak 20 Amps
Phase Current (peak)	-	60	Amps	Max 60amps capacity for fluctuation.
Logic Input Voltage (PWM & DIR)	0	5	Volts	Can be given through pulse generator or some microcontroller.
PWM Frequency	0	20	KHz	Output frequency will be same as input frequency.

Note : Peak current is limited by the over-current protection circuit. Actual current limit is depending on board temperature. Value shown is at room temperature (25 - 30 degree Celsius).

Pin Configuration of the Drive:

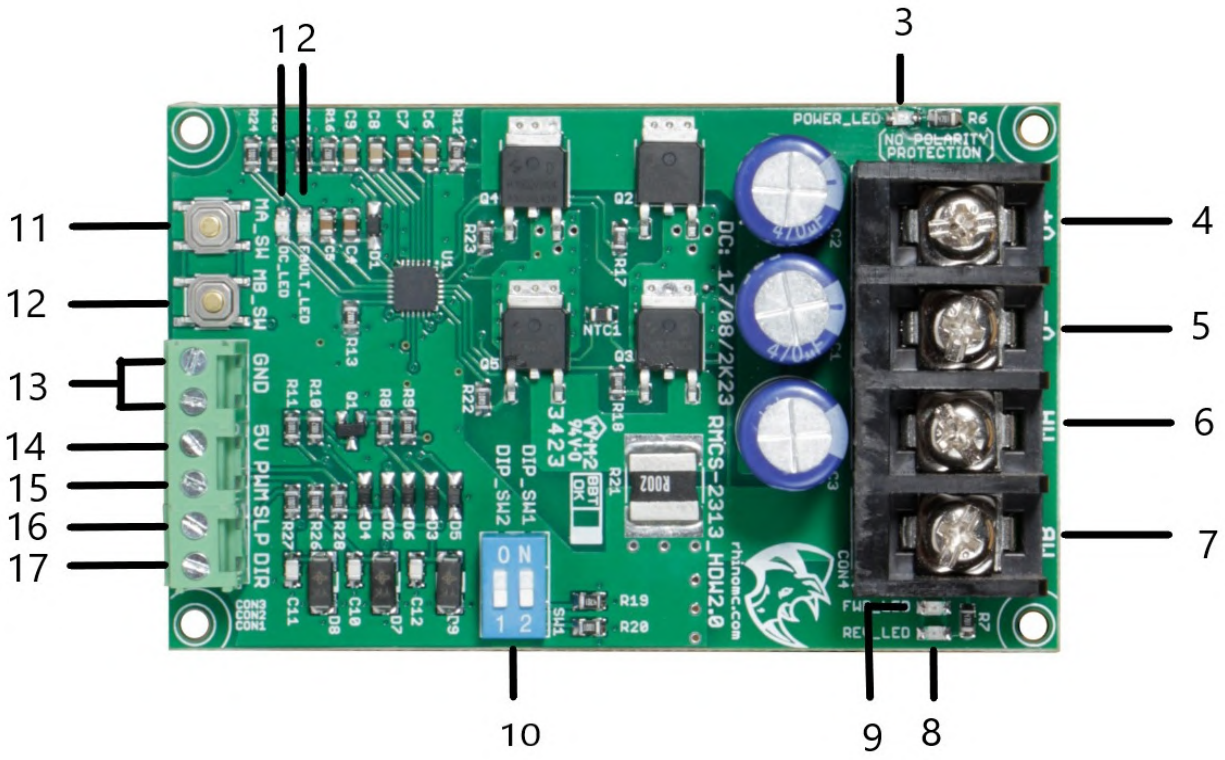


Fig 1:

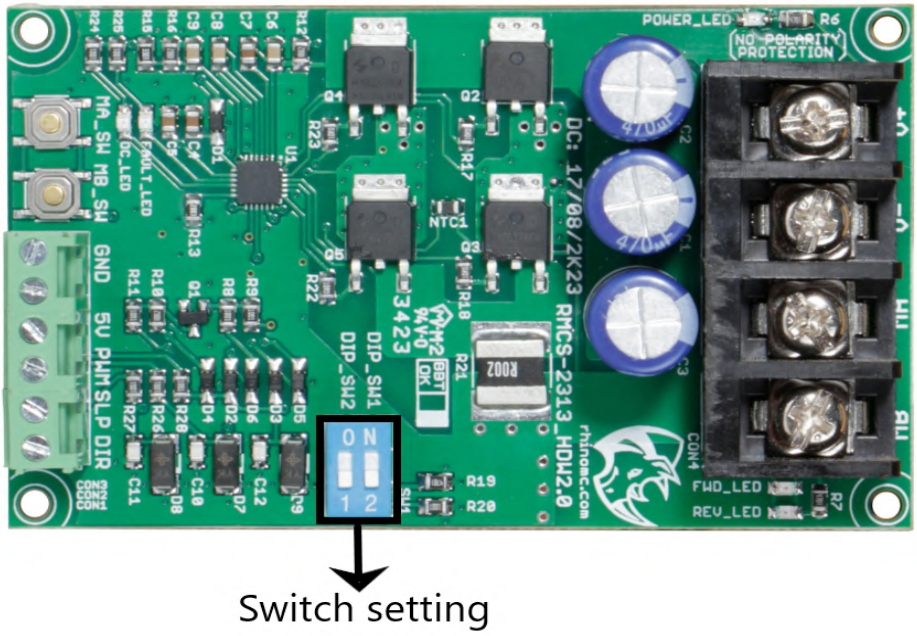
No	Description
1	OC_LED
2	FAULT_LED
3	POWER_LED
4	VCC (POWER SUPPLY)
5	GND (POWER SUPPLY)
6	MA
7	MB
8	REV_LED
9	FWD_LED

No	Description
10	CURR_SWITCH
11	MA_SW
12	MB_SW
13	GND
14	5V
15	PWM
16	SLP
17	DIR

NAME	FUNCTION
OC_LED	Turn on when current limiting is in action. Current limit threshold is depending on the board temperature.
FAULT_LED	Turn on during Under voltage, Shutdown or Hardware fault
POWER_LED	Indicates power is on. If its not on then check your connections.
VCC	Connect positive of power supply. (6 – 30 V)
GND	Connect GND of power supply.
MA	Connect motor terminal.
MB	Connect motor terminal. (Connect in reverse to change direction.)
REV_LED	Turns on when motor is in reverse direction. (Motor's direction will be dependent on the motor connection.)
FWD_LED	Turns on when motor is in forward direction. (Motor's direction will be dependent on the motor connection.)
CURR_SWITCH	Switch to limit current output.
MA_SW	Switch to run motor clockwise.
MB_SW	Switch to run motor counter clockwise.
GND	Connect with ground of microcontroller.
5V	+5V output.
PWM	PWM input for motor.
SLP	Active low. Motor Enable (Motor Free).(Can give through switch from GND pin or microcontroller pins.)
DIR	Use to change the direction of motor. (Can give through switch from +5V pin or microcontroller pins.)

Switch Settings of the Drive:

Fig 2:

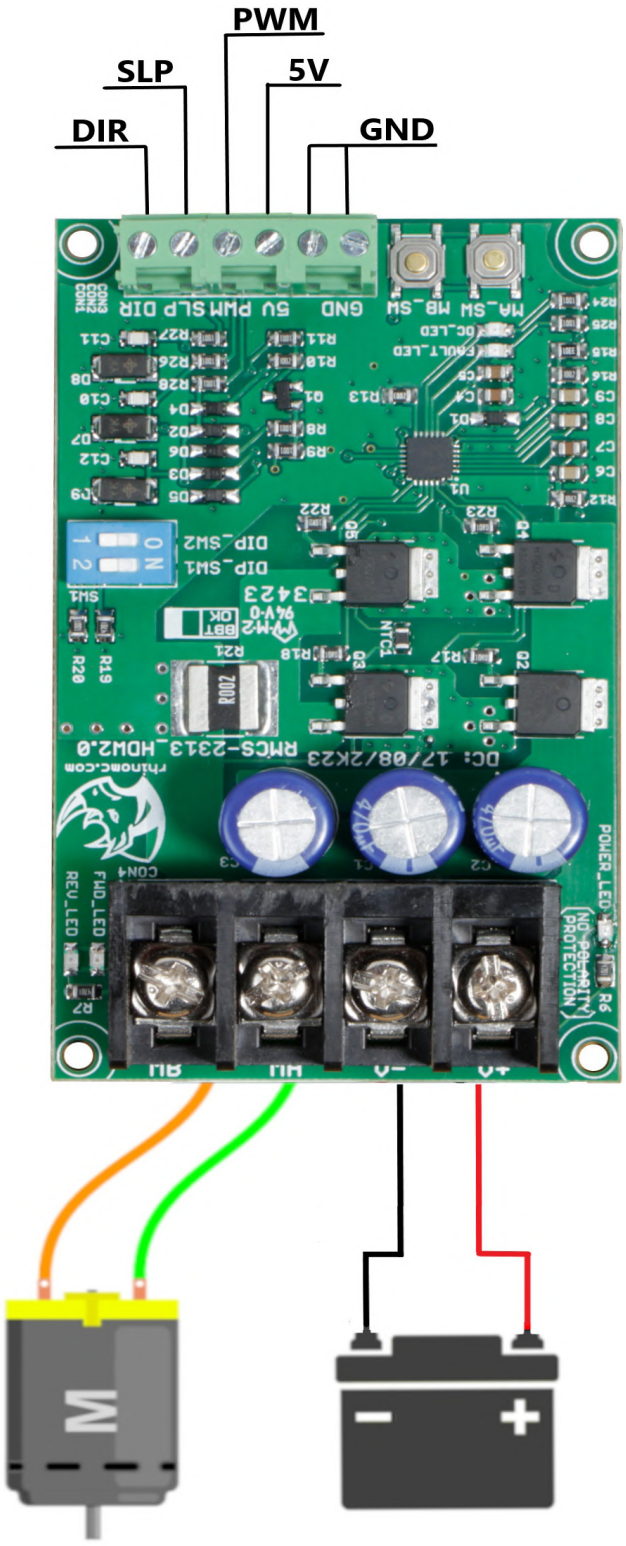


Peak current Motor M1	Switch S1	Switch S2
60 Amps	OFF	OFF
40 Amps	ON	OFF
16 Amps	ON	ON

Connection Diagram :

Hardware Connection :

Fig 3:



Controlling :

As depicted in fig 3, there is a provision to connect a motor to the driver :

To initiate a test run of the motors, you can conveniently utilize the push buttons integrated into the driver, specifically MA_SW and MB_SW. It is important to note that the direction of each motor is determined by its individual connection. Reversing the motor's connection will result in a reversal of its direction of rotation.

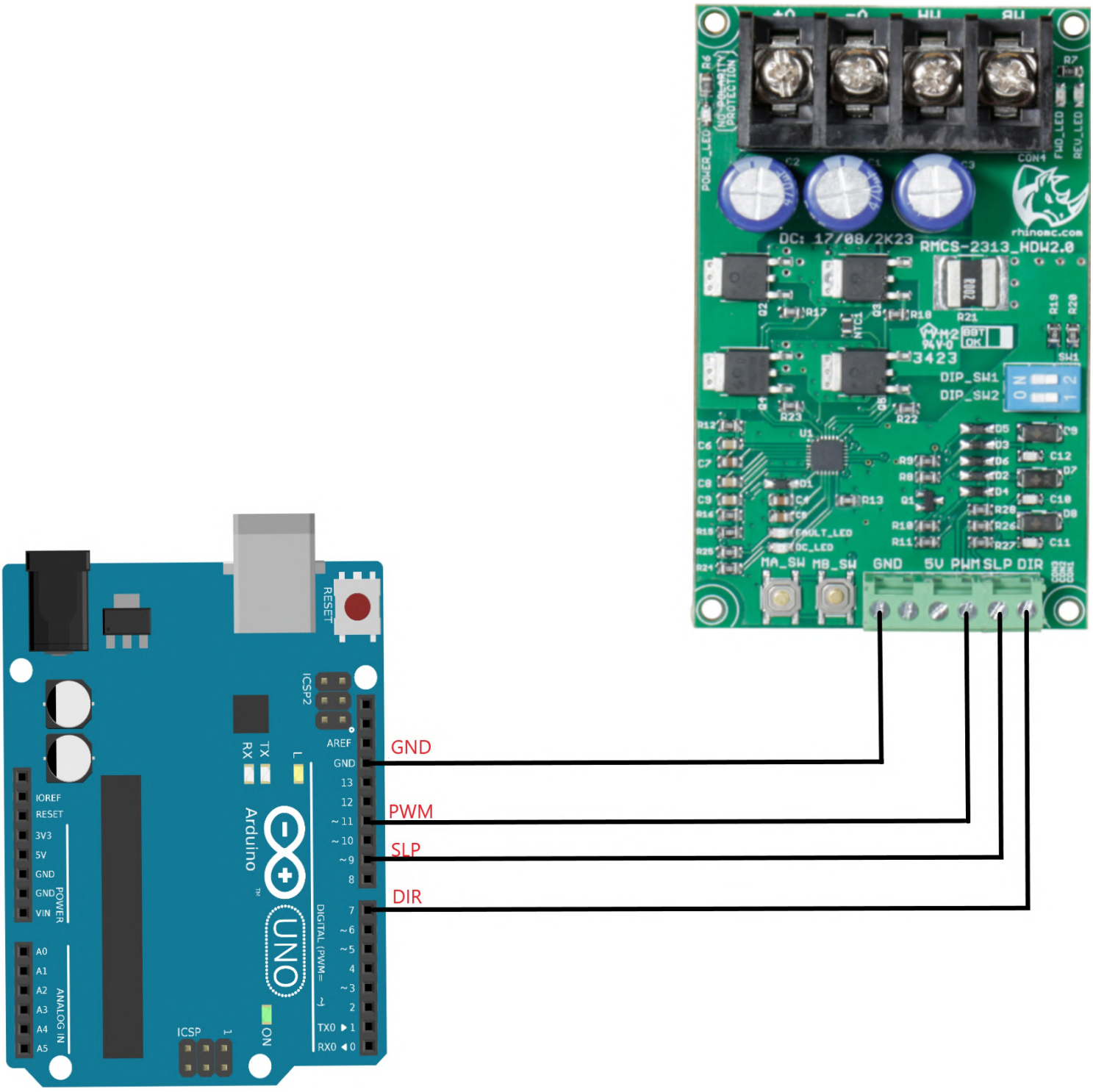
Motor control using Microcontroller :

The motors can be easily controlled using any microcontroller equipped with 5V pins. By providing PWM signals to the PWM pin of the drive, the motor speed can be adjusted accordingly.
To change the direction of the motors, a signal from any digital 5V pin can be directed to the corresponding direction pin. Furthermore, to stop the motor motion, a digital low signal from any digital pin can be applied to the sleep pin of the drive. For a visual representation of these connections, please refer to fig 4 on the subsequent page.

Motor control using Potentiometer :

If you intend to control the motor's speed using a potentiometer, you will need to employ a microcontroller such as Arduino. This microcontroller can receive input from the potentiometer and subsequently provide output to the PWM pins of the drive. For a visual reference regarding this setup, please refer to fig 5 on page 11.

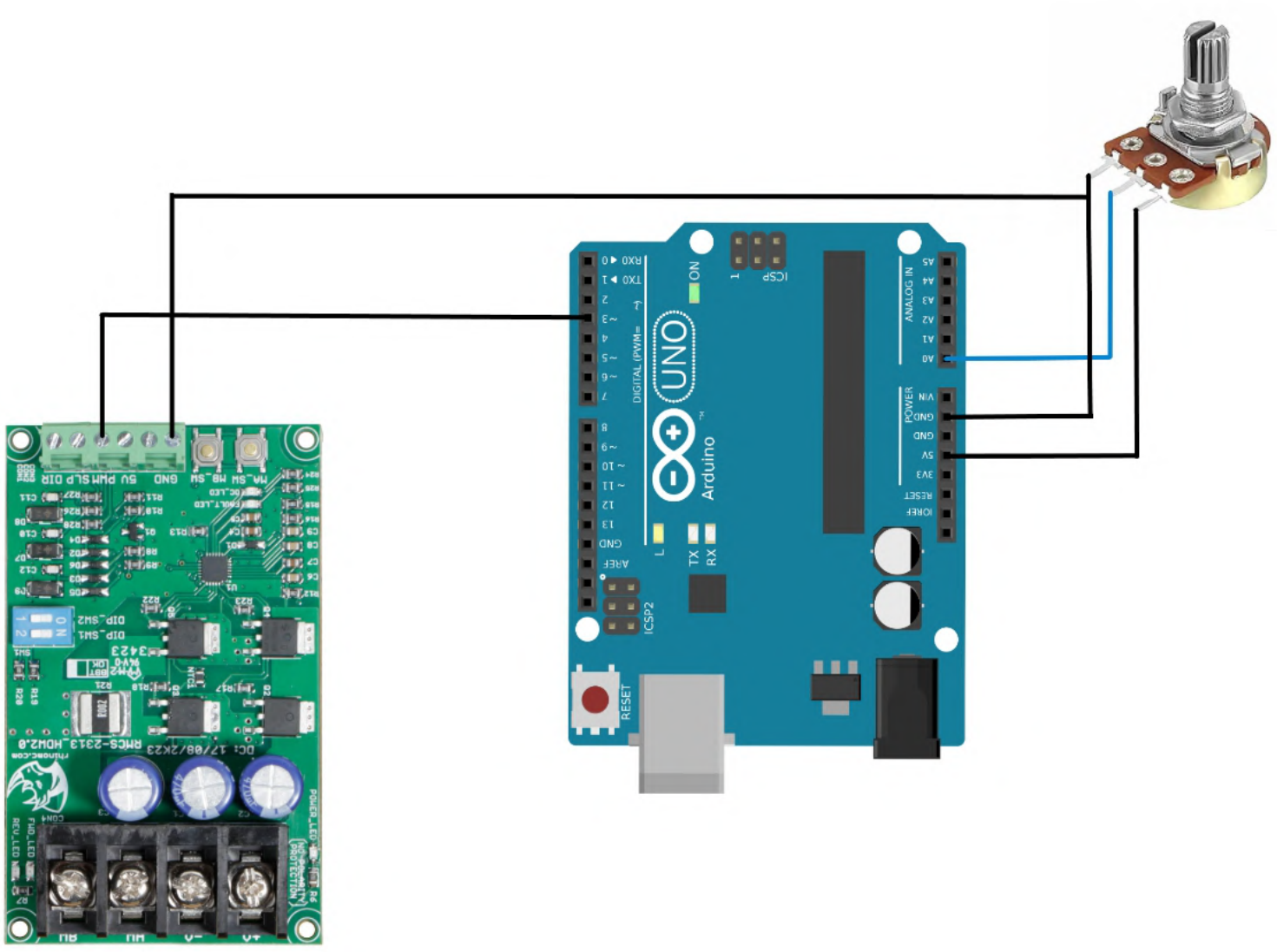
Fig 4:


GND
PWM
SLP
DIR
MA_SW MB_SW GND 5V PWM SLP DIR
DC: 17/08/2K23 RMCS-2313_HDW2.0
DIP_SW1
DIP_SW2
Arduino
UNO

Sample code for motor control using Arduino :

```
/* Connections of RMCS-2313 with arduino :
 *  RMCS2313 : Arduino pins
 *  Sleep : 9
 *  GND : GND
 *  Dir : 7
 *  pwm : 11 */

const int Sleep = 9; //Sleep pin for the motor
const int Dir = 7; //pin to change direction for motor
const int pwm = 11; //Pwm pin for motor



void setup()
{

  pinMode(Sleep,OUTPUT);
  pinMode(Dir,OUTPUT);
  pinMode(pwm,OUTPUT);

}

void loop() {


  digitalWrite(Sleep,HIGH); // To stop motor change Signal to LOW
  digitalWrite(Dir,HIGH);  // To change the direction of motor, change signal to LOW

  analogWrite(pwm,255); // you can value in between 0 - 255


}
```

Fig 5 :


UNO
Arduino
ON
L
TX
RX
RESET
POWER
ANALOG IN
DIGITAL (PWM=)
ICSP
ICSP2
5V
GND
VIN
A0
A1
A2
A3
A4
A5

Sample code for speed control using potentiomer :

Here you can check this code of Arduino to run motor with potentiomer and Arduino :

```
//Constants:
const int PwmPin = 3;  //pin 3 has PWM funtion
const int potPin = A0; //pin A0 to read analog input

//Variables:
int value; //save analog value

void setup(){
  //Input or output?
  pinMode(PwmPin, OUTPUT);
  pinMode(potPin, INPUT);
}

void loop(){
  value = analogRead(potPin);          //Read and save analog value from potentiometer
  value = map(value, 0, 1023, 0, 255); //Map value 0-1023 to 0-255 (PWM)
  analogWrite(PwmPin, value);          //Send PWM value to drive
  delay(100);                          //Small delay
}
```