ARC-S User Manual

Integrated Arduino Controlled Stepper Driver

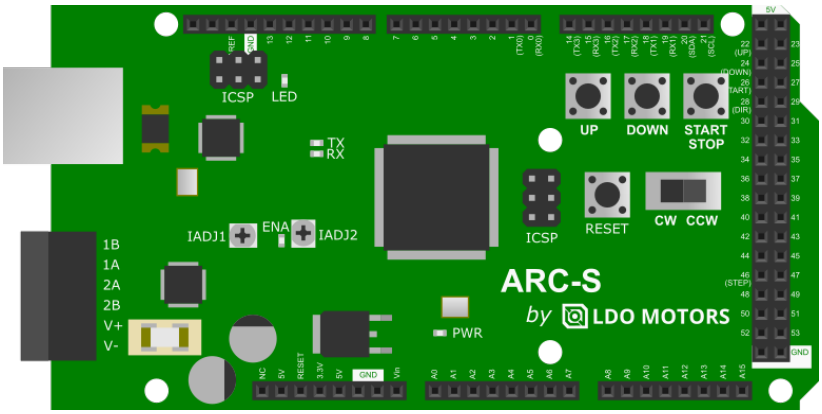
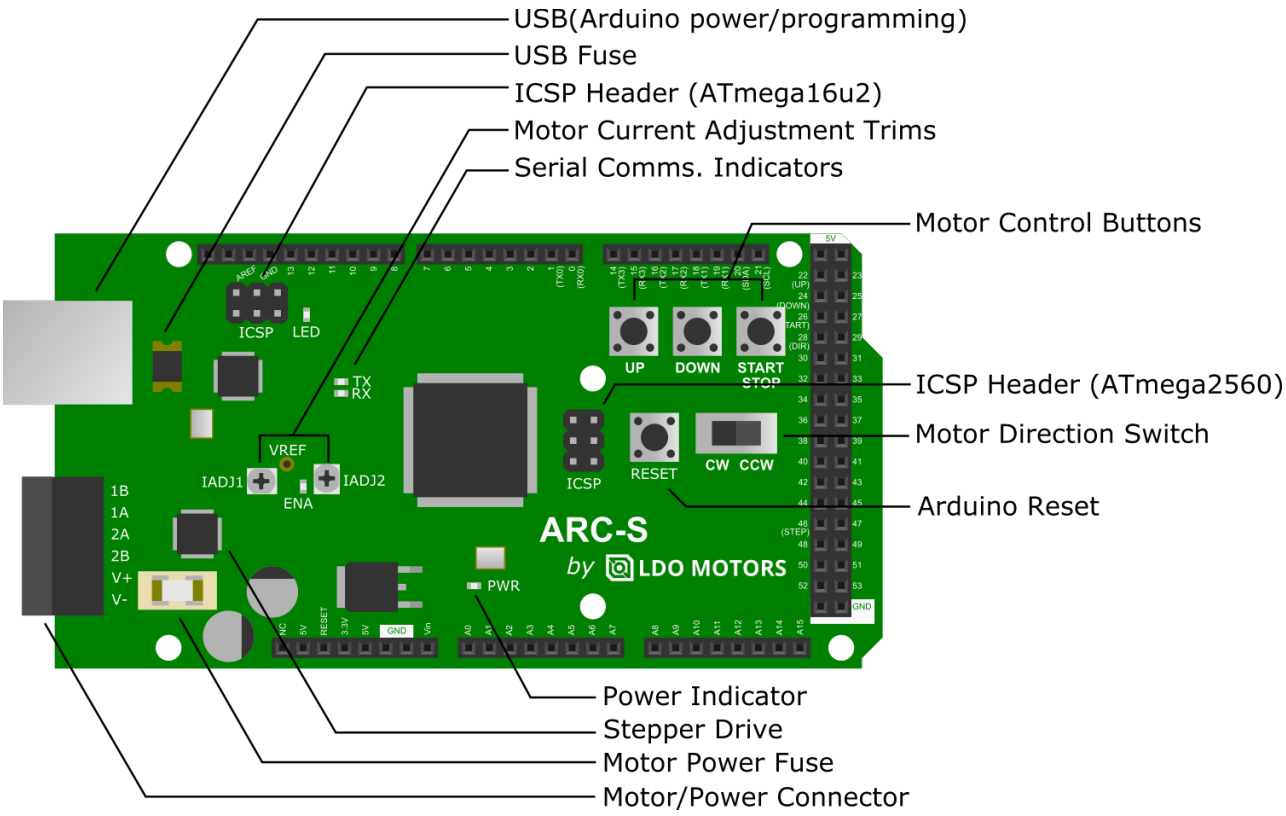


Table of Contents

1. Overview

The ARC-S (ARduino Controller – Stepper) is an Arduino[1] based microcontroller board with an integrated stepper drive. ARC-S combines the powerful convenience of the Arduino platform with the proven Allegro[2] A4988 stepper motor driver. In addition, built-in push buttons on the board can be used as a simple control panel. ARC-S comes with open source sample code and is perfect for proof-of-concept, rapid prototyping, DIY projects and more. ARC-S is plug and play - a DC power supply, stepper motor and ARC-S board is all that's needed to get your project running.



[1] Arduino® and other Arduino brands and logos are trademarks of Arduino AG.

[2] Allegro® is a registered trademark of Allegro MicroSystems, LLC.

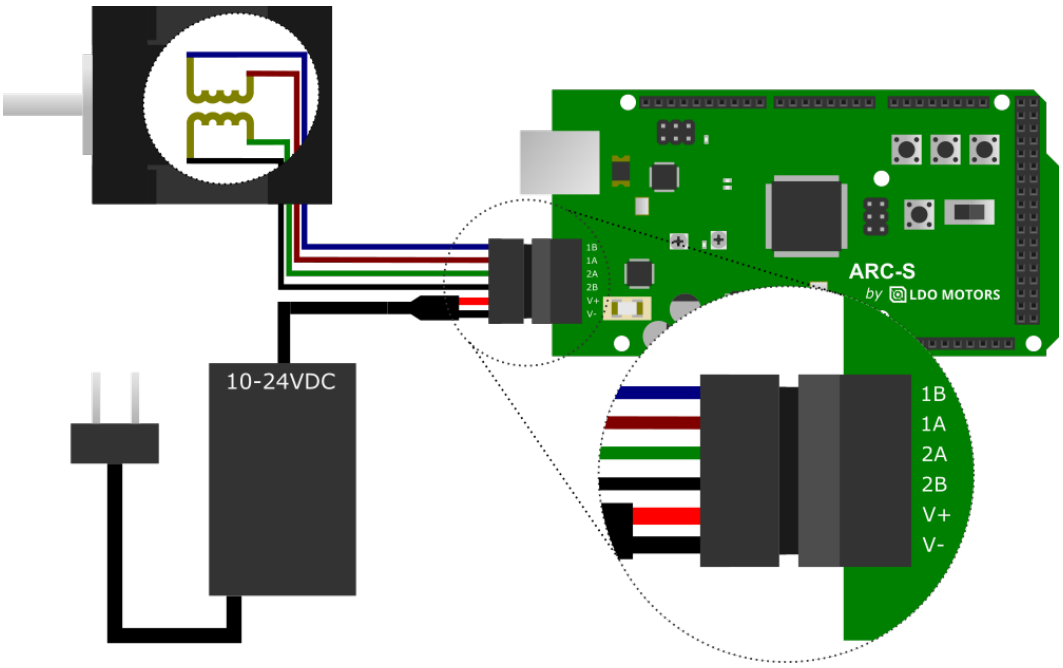
2. Specifications

Microcontroller	ATmega2560
Operating Voltage (Microcontroller)	5V
Max DC Current per Pin	20mA
Operating Voltage (Stepper Driver)	10V-24V
Max Output Current (Stepper Driver)	2A
Microstep Resolution	Full, 1/2, 1/4, 1/8, 1/16
GPIO Pins	53
ADC Pins	16
Flash Memory	256KB
SRAM	8KB
EEPROM	4KB
Clock Speed	16Mhz
Dimensions (L*W*H)	113 x 53.3 x 19mm
Weight	45g

3. Getting Started

Only three items are needed to get your stepper motor running:

- Stepper Motor.
- ARC-S board.
- DC adapter/power supply.



3.1. Power Supply Connection

ARC-S accepts DC power supplies with outputs from 10-24V. The positive and negative wires of the DC power supply connect to V+ and V- on the terminal block respectively.

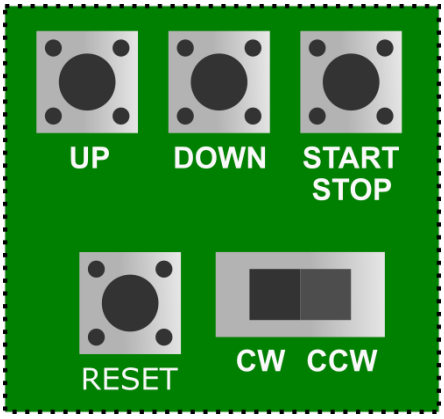
3.2. Stepper Motor Connection

ARC-S is designed to drive 2 phase bipolar stepper motors. The four wires on a stepper motor correspond to two phases, each with two wires. Connect one pair of wires from one phase to 1A and 1B; connect the other pair of wires from the other phase to 2A and 2B.

To identify which pairs of wires are within the same phase, use the continuity function (diode or beep symbol) on any multimeter. Wires within the same phase will be identified by the multimeter as shorted. The order of the wiring within a phase pair is usually not critical, and only affects the default direction in which the motor is turned. This direction can be reversed by swapping the order of wires within a ***single*** phase.

3.3. Basic Operation

ARC-S comes shipped with a speed control program downloaded onto the Arduino. With after finishing wiring, simply press the START/STOP button to start the stepper motor. The stepper motor will run continuously until the START/STOP button is pressed again. To adjust the speed of the motor, press the UP or DOWN button. Speed can be adjusted while the motor is still or moving. To change direction, toggle the direction switch labelled CW/CCW.



4. Sample Code and Library

ARC-S comes shipped with sample code pre-programmed into the microcontroller. The default code allows the user to start or stop the motor, change its direction, and adjust its speed. Other sample code and a free for use open source library is provided on
Github: https://github.com/MotorDynamicsLab/arduino-controller-stepper

The ARC-S can be reprogrammed multiple times and can be used as a normal Arduino Mega board. The lightweight Arduino IDE can be used to program the ARC-S board (identified in the IDE as Arduino Mega 2560). The IDE, along with extensive documentation is available from the official Arduino website:

IDE - https://www.arduino.cc/en/Main/Software

Documentation - https://www.arduino.cc/reference/en/

5. Power

The microcontroller in ARC-S can be powered with either the USB cable or DC power supply (V+/- terminals).

5.1. USB Power

A USB only connection only provides 5V power, which is sufficient for the basic Arduino functionality but not capable of powering the stepper motor driver. The ARC-S can be programmed as an ordinary Arduino Mega with the USB cable connected to a computer.

5.2. DC Power Supply

A DC power supply can supply power to both the Arduino and Stepper Driver. ARC-S accepts voltage ranges of 10 – 24V from the V+ and V- terminals. A power supply with an output of 2A or higher is recommended.

When using DC power supply, the Arduino microcontroller will derive its 5V power from the onboard 78M05 voltage regulator - capable of supplying up to 1A of current. The stepper driver will use whatever voltage level from the power supply to power the stepper motor. Using a higher voltage typically allows the stepper motor to achieve higher speeds.

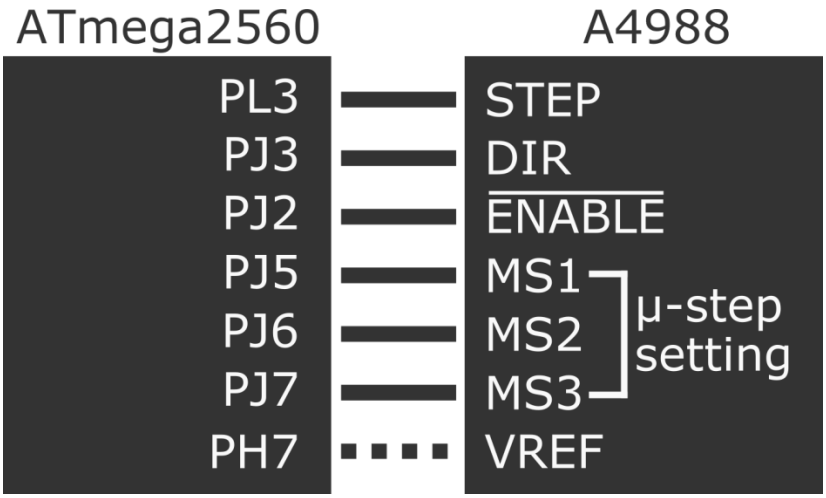
5.3. Fuses

Each of the USB and DC power supply circuits contain a fuse. The fuse used for USB power is a 0.5A resettable fuse. The fuse used in the DC power supply circuit is a replaceable 3A Nano 2 fuse. The DC power supply fuse is mounted on a fuse holder - allowing it to be replaced easily in the event of a blown fuse.

6. Stepper Driver Interface

The ATmega2560 microcontroller from the Arduino portion of ARC-S interfaces with A4988 stepper drive through the following pins: STEP, DIR, ENABLE, MS1, MS2, MS3, and VREF.

STEP, DIR, ENABLE, MS1, MS2, MS3 are discrete input pins with a direct connection to the microcontroller. The VREF pin on the A4988 is an analog input and is controlled indirectly by the microcontroller (see **7.3 Stepper Driver Current Setting**)



6.1. Enable Pin

The ENABLE pin in essence, switches the stepper drive on or off. The polarity of the pin is active low, so the stepper drive is enabled when the pin is logic low and is disabled when the pin is logic high. The following summarizes the behavior of the stepper drive and stepper motor when in the Enable and Disabled state. When the enable pin is not configured, a pull-up circuit on the PCB ensures that the stepper driver stays disabled by default.

	Stepper Drive (A4988)	Stepper Motor
Enabled (Logic LOW)	• Constantly draws current • Chip will heat up • Accepts (Step/Direction) movement commands	• Motor frame heats up regardless of movement. • Shaft is locked when not commanded to move
Disabled (Logic High)	• No current draw, no heat • Ignores movement commands	• Unpowered, no heat • Shaft is free to move

6.2. Step and Direction Pins

The STEP and DIR pins control the movement of the stepper motor.

The stepper drive moves the motor by **one step** when the STEP pin transitions from logic low to logic high (a.k.a. rising edge). The size of each step is controlled by the Microstep settings explained in **6.3 Microstep Setting**. Thus the user can control the distance in which the motor moves by controlling the number of square wave pulses on the STEP pin as well as the speed of the motor by varying the frequency of the squarewave.

The DIR pin controls the direction of the stepping motion. Changing the logic level on the DIR pin will change the direction of movement from subsequent STEP signals.

6.3. Microstep Setting

The combined logic value of pins MS1, MS2, MS3 determines the microstep setting of the stepper drive. The following table defines possible combinations of MS1, MS2, MS3 and their corresponding microstep setting:

MS1	MS2	MS3	Microstep Resolution	Steps/Rev (on 200 step motor)
LOW	LOW	LOW	Full Step (No microstep)	200 steps/rev
HIGH	LOW	LOW	Half Step	400 steps/rev
LOW	HIGH	LOW	1/4 Step	800 steps/rev
HIGH	HIGH	LOW	1/8 Step	1600 steps/rev
HIGH	HIGH	HIGH	1/16 Step	3200 steps/rev

Choosing to use microstepping can provide a number of benefits apart from smaller step sizes:

- Even when using a 200 step/rev stepper motor, spinning the motor at full steps generates a large amount of mechanical vibration and noise. Using higher resolution microstepping can reduce this vibration greatly.

- Using a suitable microstep can allow the stepper motor to achieve much higher speeds than that which would be possible on full steps.
- One important drawback of microstepping to note is that using higher microstep settings can significantly reduce the incremental torque of the motor. In other words, microstepping with too high a resolution can result in steps being skipped/missed due to low torque.

6.4. Pin Manipulation

ARC-S uses pins of the ATmega2560 chip which are normally not connected on an ordinary Arduino Mega to interface with the stepper driver chip (except STEP, which is connected to D46):

A4988	STEP	DIR	ENABLE	MS1	MS2	MS3	VREF (switch)
ATmega2560	PL3 (D46)	PJ3	PJ2	PJ5	PJ6	PJ7	PH7

The normal Arduino functions **`pinMode()`** and **`digitalWrite()`** do not apply to these pins.

To setup a pin as ouput use:

```
DDRx |= (1<<y);
```

where *x* is the port and *y* is the pin offset. For example, setting up ENABLE pin:

```
DDRJ |= (1<<2);
```

To set a pin to logic high use:

```
PORTx |= (1<<y);
```

For example, setting STEP to logic high:

```
PORTL |= (1<<3);
```

Similarly, to clear a pin to logic low use:

```
PORTx &= (~(1<<y));
```

For example, clearing MS1:

```
PORTJ &= (~(1<<5));
```

For more information on direct pin manipulation
see: https://www.arduino.cc/en/Reference/PortManipulation

7. Stepper Current Control

Unlike typical motors, the current draw of stepper motors has very little to do with motor speed, load, or voltage. Stepper motor drivers typically provide an interface to control the maximum current flow. It is

important to understand this current setting in order to power your stepper motor as efficiently as possible.

7.1. Rated Current

Stepper motor specifications will usually supply a **rated current** value. This rated current value denotes the maximum amount of current that can be allowed to flow through the stepper motor for long periods of time that would not cause damage to the motor. While it may be tempting simply configure the current setting on the stepper driver to this rated current value, it is almost always recommended to use a lower current setting (see the next section).

Another value that is often seen in motor specification sheets is the rated voltage. This value is typically much lower than what one might expect. rated voltage is simply the product of coil resistance and rated current. Components in the stepper driver and stepper motor are usually damaged by heat - heat is a function of current and **not voltage**. Due to the fact that the stepper driver is used to regulate current regardless of supply voltage, rated voltage can be safely ignored.

7.2. Current vs. Motor Performance

Like other motors, the current flowing through a stepper motor influences the amount of torque it generates. Unlike other motors, increasing current does not impact the speed of the stepper motor. One way to interpret this is that stepper drivers control position rather than speed.

A stepper driver that is enabled will constantly draw a fairly constant level of current regardless of the motor's speed. When choosing a suitable motor current for a given application, the most important consideration is the mechanical load on the motor. Users should experiment with different current levels and choose the lowest current that is capable of driving their mechanical load.

7.3. Stepper Driver Current Setting

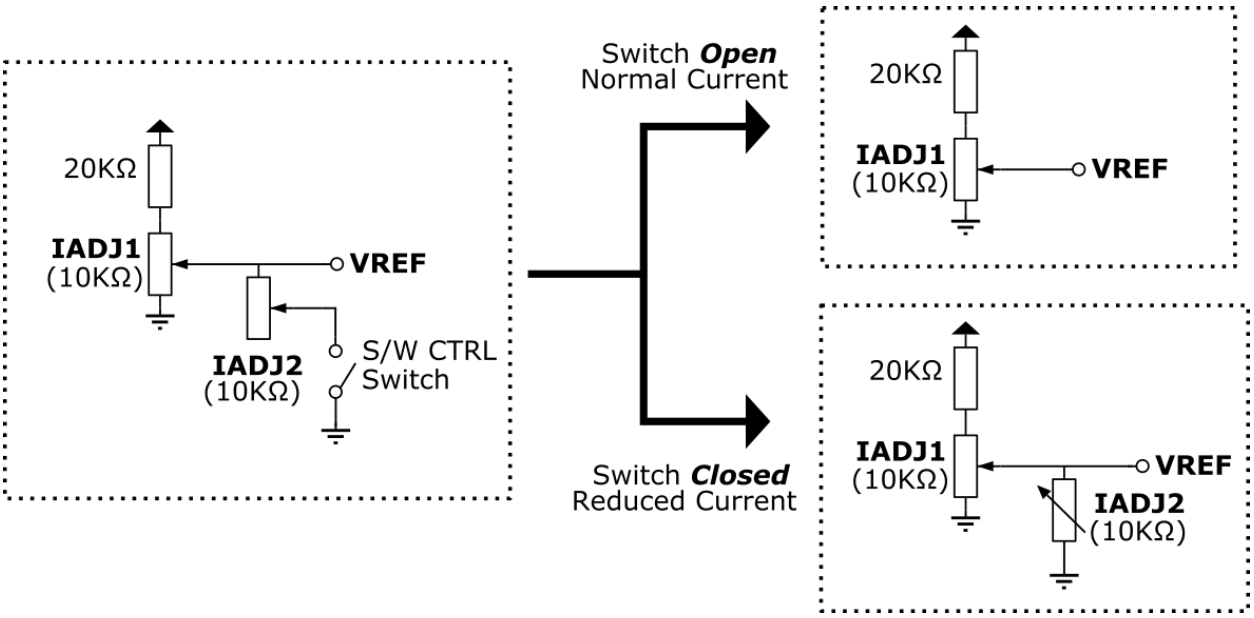
Current flowing through the stepper motor is configured by varying the voltage level of the pin VREF on the stepper driver. The relationship between motor current and VREF is as follows:

$$I_{motor} = \frac{V_{ref}}{8 \cdot R_s} = \frac{V_{ref}}{0.8}$$

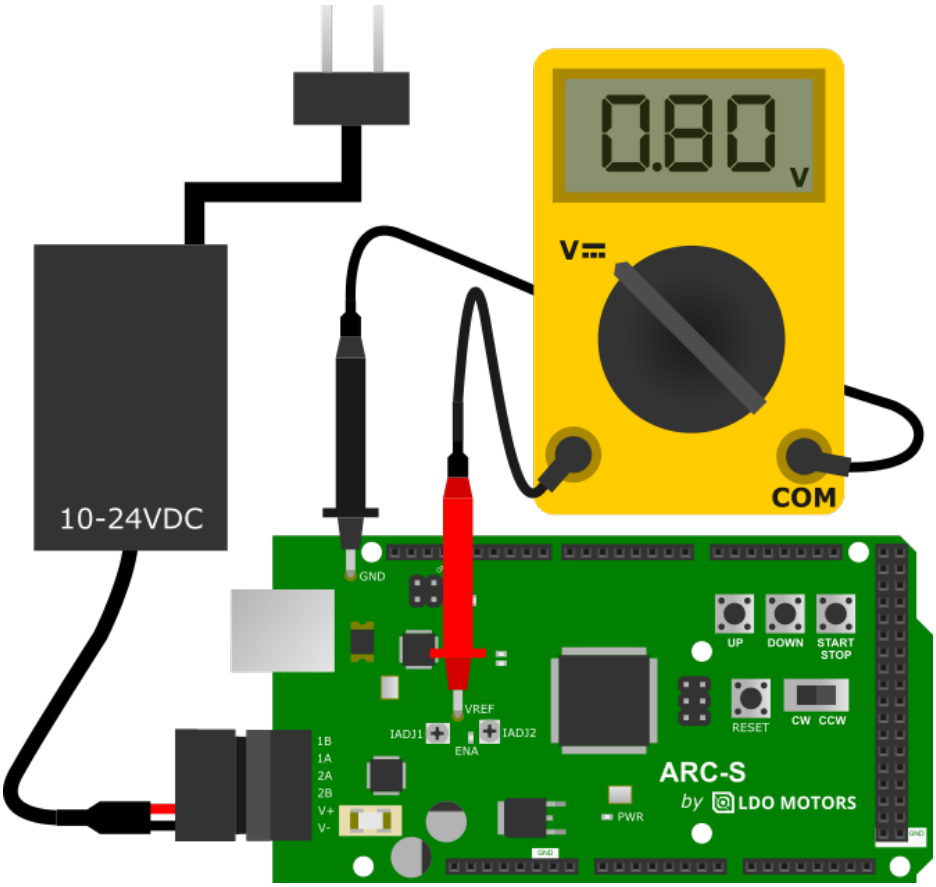
Where R_s = 0.1Ω is the value of the current sense resistor. By rearranging the equation, we can determine the VREF required to drive a given motor current:

$$V_{ref} = 0.8 \cdot I_{motor}$$

The voltage level of VREF can be adjusted using the two trimpots **IADJ1** and **IADJ2**. The via labeled VREF allows the voltage to be measured easily using a multimeter. A switch (PH7 on the Arduino) allows the user to switch between normal current mode (adjusted solely using IADJ1) and low current mode (adjusted using both IADJ1 and IADJ2).



7.4. Procedure for Adjusting Motor Current



1. Program the ARC-S board with ***current-setting.ino*** from the online code sample.
2. Connect the ARC-S board to the power supply. Set a multimeter in DC voltage mode, connect the positive lead to the via labelled **VREF** and the negative lead to any ground.
3. Calculate the voltage needed on VREF from the equation in 7.3 Stepper Driver Current Setting.
4. Adjust the trimpot labelled **IADJ1** until the reading on the multimeter matches the voltage calculated from step 3.
5. *OPTIONAL*, press **DOWN** to switch to "low current" mode and repeat steps 3-4, using trimport **IADJ2** and a newly calculated voltage.
6. *OPTIONAL*, connect a motor. Press **START/STOP** to enable the stepper driver. Use a multimeter in series with one of the motor leads to verify the actual motor current.

8. Simple Control Panel

The control panel on ARC-S contains 3 momentary pushbuttons (UP, DOWN, START/STOP) and a single switch (CW/CCW). These buttons and switches are linked to normal Arduino digital I/O pins and can be read using `digitalRead()`.

Button	Arduino Pin
UP	22
DOWN	24
START/STOP	26
CW/CCW	28

For the pushbuttons (UP, DOWN, START/STOP), pressing the button connects the corresponding Arduino pin to ground (logic LOW); releasing the button leaves the pin unconnected (floating).

For the switch (CW/CCW), toggling to "CCW" connects the Arduino pin to ground while toggling to "CW" leaves the pin unconnected.

Control panel pins should be configured using `pinMode(x, INPUT_PULLUP)` before being used. This enables the internal pull-up and makes the pins read logic high when they are unconnected externally.

9. Heat Dissipation

The heat generated by the onboard stepper driver chip is directly proportional to the current setting of the driver. At higher currents, the hot temperature on the driver chip can reduce the lifespan or even cause a thermal shutdown. ARC-S dissipates heat generated from the driver chip by using the PCB bottom copper plane as a passive heatsink. In addition, a large aluminum heatsink is provided, which can be attached from the button of the PCB to provide additional heat dissipation.

10. Arduino Compatibility

The ARC-S is designed to be as compatible with the Arduino Mega as possible. The PCB outline of ARC-S is identical to the Arduino Mega Rev3. The USB socket, I/O pin headers, ICSP headers and reset switch are also in identical positions to the Arduino Mega. When not used for controlling stepper motors, the ARC-S can be used as an ordinary Arduino Mega microcontroller board. The following lists differences between the ARC-S and an Arduino Mega:

- Digital I/O pin 46 is used by the stepper driver as the STEP signal. It is connected to a 10kΩ pull-up resistor.
- Digital I/O pins 22, 24, 26 are connected to momentary pushbuttons used by the simple control panel. As long as these buttons are not pressed, these pins can be considered not connected. When pressed, they are connected directly to ground.
- Digital I/O pin 28 is connected to a toggle switch used by the simple control panel. As long as the switch is in the "CW" position, the pin can be considered not connected. Toggling the switch to the "CCW" position connects the pin directly to ground.
- The original 2.1mm power jack is replaced with a 6 pin connector to accommodate the stepper motor and DC power in. The original power jack only accepts input voltages up to 12V whereas ARC-S will accept voltages of up to 24V.