Stepstick a4988 datasheet

In this Arduino Tutorial we will learn how to control stepper motor using A4988 Stepper Driver. You can watch the video below or read the written tutorial below. The A4988 is a micro-slice controller for controlling bipolar stepper motors, which has a built-in translator for easy operation. This means that we can control the stepper motor with just 2 pins from our controller or one to control the direction of rotation, and the other to control the stairs. The driver provides five different step resolutions: full step, embroidery-step, quarter-step, eight-step and sixteenth-step. In addition, it has a potentiometer to regulate the output power of the current, thermal shutdown over temperature and protection of the return current. Its logical voltage is between 3 and 5.5 V and the maximum current per phase is 2A if good additional cooling or 1A continuous current per phase is provided without heat sink or cooling. A4988 Stepper Driver Pinout Now let's look closely at the driver pinout and connect it with stepper motor and controller. So we will start with 2 pins on the right side of the button to power the controller, VDD and ground pins, which we need to connect them to the power supply 3 to 5.5 V, and in our case it will be our controller, Arduino board, which will provide 5 V. The following 4 pins are to connect the motor. Pins 1A and 1B will be connected to one motor coil, and pins 2A and 2B to the other motor coil. To power the engine we use another 2 pins, Ground and VMOT, which we need to connect to the power supply from 8 to 35 V, and we also need to use a separating capacitor from at least 47 µF to protect the driver's board from voltage spikes. The next two 2 pins, Step and Direction are the pins we use to control motor movements. Pin Direction controls the direction of rotation of the motor and we need to connect it to one of the digital pins on our microcontroller, or in our case I will connect it to the pin number 4 of my Arduino board. With the Step pin, we control the motor mirosteps and with each pulse sent to that pin, the engine moves one step. This means that we don't need any complicated programming, phase sequence tables, frequency control lines, and so on, because the built-in A4988 driver translator takes care of everything. Here we must also mention that these 2 pins are not pulled to any voltage internally, so we should not leave them floating in our program. Next is the SLEEP Pin and the low logical level puts the board to sleep mode to minimize power consumption when the engine is not in use. The RESET pin then sets the translator to a predefined native state. This home state or home microstep item can be seen from this data from the A4988 Datasheet. Therefore, these are the initial positions from which the engine starts and they differ depending on the resolution of the micro-emmeters. If the input state to this pin is low logic, all STEP inputs will be ignored. Reset Pin floating pin, so if we are not going to control it in our program, we need to connect it to the SLEEP pin to bring it high and turn on the board. The next 3 pins (MS1, MS2, and MS3) are used to select one of the five step resolutions according to the truth table above. These pins have internal resistors extended, so if you leave them disconnected, the board will work in full step mode. The last ENABLE pin is used to turn FET outputs on or off. Thus, high logic will keep the output disabled. Components needed for this Arduino Tutorial You can get components from any of the following sites: Disclosure: These are affiliate links. As an Amazon Associate, I earn money from eligible purchases. Circuit diagrams Here are the full circuit diagrams. I'll use the drive in full step mode, so I'll leave the 3 ms pins disconnected and just plug the direction and step pins of the drive into pins number 3 and 4 on the Arduino board, as well as the ground and 5 V pins to power the board. Also I will use a 100µF capacitor for separation and a 12V, 1.5A adapter to power the motor. I will use the NEMA 17 bipolar stepper motor and its A and C wires will be connected to pins 1A and 1B and B and D wires up to 2A and 2B pins. Current limitation Before connecting the engine, we should adjust the current driver limitation so that we can be sure that the current is within the current limits of the engine. We can do this by adjusting the reference voltage using the potentiometer on the board and considering this equation: Current limit = VRef x 2 However, this equation is not always correct because there are different manufacturers of the A4988 controller board. Here's a demonstration of my case: I adjusted the potentiometer and measured the reference voltage of 0.6V. Thus, the current limit should be 0.6 *2, equal to 1.2 A. Now, since I use the controller in full step mode and according to the A4988 Datasheet in this mode, the winding current can reach only 70% of the current limit, 1.2A * 0.7 will be equal to 0.84A. To check this, I sent a simple code that sends continuous logic high to the Step pin (so we can better notice the current) and connected my meter in series with one winding motor and powered it. What I got was 0.5A, which means that the equation was not correct for my case. Here's a sample code. First we need to define the step and direction pins. In our case, these are pins number 3 and 4 on board Arduino and they are named stepPin and dirPin and the configuration section we need to define them as outputs. Step motor control exmaza code * * dejan Nedelkovski, www.HowToMechatronics.com * */ // defines pin numbers const int stepPin = 3; const int dirPin = 4; void setup() { // Sets two pins as pinMode(stepPin,OUTPUT); pinMode(dirPin,OUTPUT); } void loop() { digitalWrite(dirPin,HIGH); // Allows the engine to move in a specific direction // Makes it pulses to perform one full cycle rotation for (int x = 0; x &lt; 200; x++) { digitalWrite(stepPin,HIGH); delayMicroseconds(500); digitalWrite(stepPin,LOW); delayMicroseconds(500); delay(1000); One second delay digitalWrite (dirPin,LOW); Changes the direction of rotation // Makes 400 pulses to perform two full cycle revolutions for (int x = 0; x &lt; 400; x++) { digitalWrite(stepPin,HIGH); delayMicroseconds(500); digitalWrite(stepPin,LOW); delayMicroseconds(500); } delay(1000); } First, in the loop section, we will set the root to a high state that will allow the engine to move in a certain direction. Now using this to loop, we will make the engine do one full cycle turn. Since the controller is set in full step mode and our stepper motor has a step angle of 1.8 degrees or 200 degrees, we need to send 200 pulses to the access to perform one full cycle rotation. So the for loop will have 200 iterations and each time it will set the Step pin to a high and then low state to perform pulses. Between each digitalWrite we need to add some delay, on which the engine speed will depend. After this full rotation cycle, we will make one second delay, and then change the direction of rotation by setting the dirPin to a low state, and now perform 2 full cycle revolutions with this loop of 400 iterations. Finally, there is one more second delay. Now let's submit the code and let's see how it will work. I made one more example for this tutorial where I can control the engine speed with a potentiometer. Here is the source code for this example: /* Simple Stepper Motor Control Exaple Code * * by Dejan Nedelkovski, www.HowToMechatronics.com * */ // Defines pin numbers const int stepPin = 3; const int dirPin = 4; int customDelay,customDelayMapped; Defines void setup() { // Sets two pins as pinMode(stepPin,OUTPUT); pinMode(dirPin,OUTPUT); digitalWrite(dirPin,HIGH); //Enables the motor to move in a particular direction } void loop() { customDelayMapped = speedUp(); // Gets custom delay values from the custom speedUp function // Makes the pules with custom delay, depending on potentiometer, on which the speed of the digitalWrite(stepPin, HIGH); delayMicroseconds(customDelayMapped) engine depends; digitalWrite(stepPin, LOW); delayMicroseconds(customDelayMapped); } // Int speedUp() reading function { int customDelay = analogRead(A0); // Reads the int newCustom potentiometer = map(customDelay, 0, 1023, 300,4000); // Convrests potentiometer reading values from 0 to 1023 to desired delay values (300 to 4000) return a new winner; } A4988ArduinoStepper DriverStepper MotorStepper Motor ControlTutorialRelated Posts 22 August 2019 - 0 Comments A4988 Stepper Motor Driver Module [Click image to enlarge it] A4988 is a complete Microstepping motor controller with built-in translator for easy operation. The maximum output power of the driver is 35 V ± to 2 A. with bipolar stepper motors in full, semi-, 4.4- and i Modes. Configuration Pin Name Description VDD &amp; GND Connected to 5V and GND VMOT &amp; GND Controller Used to power motor 1A, 1B, 2A, 2B Connected to 4 motor turns DIRECTION direction control pin STEP Steps Control Pin MS1, MS2, MS3 Microstep Selection Pins SLEEP Pins For Controlling Power States RESET ENABLE A4988 Stepper Driver Module Features Max. Operating Voltage: 35V Min. Operating Voltage: 8V. Max current per phase: 2A microcrone resolution: Full step, 1/2 step, 1/4 step, 1/8 and 1/16 degree Reverse voltage protection: None Dimensions: 15.5 × 20.5 mm (0.6 × 0.8) Short circuit and compact load protection Low RDS (ON) output Thermal shut-off circuits Alternative to A4988: DRV8825, L6474, L6207, L6208, TMC2208, TMC2209 Note: Full technical details can be found in the A4988 datasheet at the end of this page. How to use the A4988 driver module As mentioned earlier, the A4988 has a built-in translator, so only two wires are required to connect it to the controller board. The circuit diagram for connecting module A4988 to the microcontroller for controlling the stepper motor is shown below. As shown in the diagram above, only two DIR pins and a MODULE STEP is connected to Arduino. The STEP pin used to control the steps, while the DIR pin is used to control the direction. Micro step pins (MS1, MS2, and MS3) are used to operate the controller module in various step functions. On the above circuit, the MS1, MS2, and MS3 pins remain disconnected, which means that the driver will run in full-step mode. This motor controller has low ESR ceramic capacitors on board, making it susceptible to voltage spikes, so it is recommended to use a capacitor with a capacity of at least 47µf in the motor power pins. The stepper motor cables are connected to the output pins (1A, 1B, 2A &amp; 2B) of the controller module. It is widely used in the control of NEMA series stepper motors such as NEMA17, NEMA23 and NEMA34 Applications Used to control the speed and speed of the stepper motor. It is used in robotics to control their movement. It is used in various toys. A4988 IC 2D-Model A4988 Stepper Controller Module Engine Driver Data Sheet