MORGRITECH

Stepper Motor Control Equations

Design Development Report	dsdr1001	Issue 01	12/09/2024

Contents

1 Introduction

This document has been created as part of the development of the MT-arduino-stepper-driver library. This document (and the library) focuses on the use of stepper drivers with a "step-direction-enable" interface to control stepper motors. The use of timer interrupts is also not considered to avoid any platform/architecture specific implementation.

The library repository is located at:

https://github.com/Morgritech/MT-arduino-stepper-driver

URL 1 Link to MT-arduino-stepper-driver library repository

Upon release, the library will be made available for downloaded via the Arduino library manager:

https://www.arduino.cc/reference/en/libraries/

URL 2 Link to the Arduino libraries website

2 Definitions

Word/phrase	Description
Basic/full step	The smallest distance travelled by a stepper motor shaft when no microstepping is applied.
Microstep	The smallest distance travelled by a stepper motor shaft when microstepping is applied.
Step angle	The angle swept (usually in degrees) when a stepper motor shaft moves by a full step. This value is normally specified by the stepper motor manufacturer.
Microstep angle	The angle swept (usually in degrees) when a stepper motor shaft moves by a microstep (due to microstepping being applied).
Microstepping	The technique that allows a stepper motor to be driven in a mode whereby a full step is sub-divided into smaller steps (microsteps). This can be performed by a stepper driver.
Stepper driver	A piece of electronic equipment that drives a stepper motor (normally with the ability to perform microstepping).

Table 1 Definitions

3 Stepper motor and driver basics

The principle behind the inner workings of stepper motors and drivers is beyond the scope of this report. There is a multitude of stepper motor primers available online, the following of which are noteworthy:

https://www.orientalmotor.com/stepper-motors/technology/stepper-motor-basics.html

URL 3 Basics of Stepper Motors primer from Oriental motor (Last accessed 11/09/2024)

https://uk.rs-online.com/web/content/discovery/ideas-and-advice/stepper-motors-guide

URL 4 The Complete Guide to Stepper Motors from RS (Last accessed 11/09/2024)

4 Distance travelled

Stepper motors are controlled in steps/microsteps, and move one microstep at a time when a pulse signal is sent to the stepper driver. However, the travel distance (i.e. how far it rotates) is normally required to be set as a sweep angle (θ), normally in in S.I. (or other) units. In order to determine the travel distance/sweep angle of a stepper motor, you must first consider the step/microstep angle which depends on the microstepping mode, presented in the following section.

4.1 Microstep modes

The following modes are normally selectable on a stepper driver:

Microstep mode	Microsteps	Step angle/Microstep angle in degrees (basic step angle 1.8° as an example)
1 (Full step)	1	1.8
2 (Half step)	$^1/_2$	0.9
4 (Quarter step)	$^1/_4$	0.45
8 (Eighth step)	$^1/_8$	0.225
16 (Sixteenth step)	$^1/_{16}$	0.1125
Etc.	Etc.	Etc.

Table 2 Microstep modes

4.2 Microstep angle

Given the full step angle in degrees;

$$microstep\ angle^\circ = \frac{full\ step\ angle^\circ}{microstep\ mode}$$

Equation 1 Microstep angle

Note: If there is a gearbox in the mix, the microstep angle of the output would be further subdivided by the gear ratio, i.e.;

$$microstep\ angle^\circ = \frac{full\ step\ angle^\circ}{gear\ ratio \times microstep\ mode}$$

Equation 2 Microstep angle (with gearbox)

4.3 Sweep angle (θ) units' conversions

4.3.1 Degrees to microsteps

$$\theta_{microsteps} = \frac{\theta^{\circ}}{microstep\ angle^{\circ}}$$

Equation 3 Degrees to microsteps

4.3.2 Radians to microsteps

$$\theta_{microsteps} = \frac{180 \times \theta^{c}}{\pi \times\ microstep\ angle^{\circ}}$$

Equation 4 Radians to microsteps

4.3.3 Revolutions to microsteps

$$\theta_{microsteps} = \frac{360 \times \theta_{revolutions}}{microstep\ angle^{\circ}}$$

Equation 5 Revolutions to microsteps

5 Constant speed control

All stepper motors have a maximum speed at which they can be started without acceleration. The motor will appear to start/stop (almost) instantly. There are limitations to this method of control, including the inability of the motor to reach its maximum speed, and also the possibility of starting/stopping with a jerk effect, especially towards the top end of the maximum start speed.

Control of a stepper motor with a stepper driver involves the generation of pulse signals with a microcontroller (or otherwise), which cause the motor to move by a single step or microstep for each pulse. Hence the frequency of generation of pulses controls the speed of the motor. For the purposes of this document, the interval/period of time between each pulse is called the microstep delay/period (T), which is presented in the following section.

5.1 Microstep period (T)

This is essentially the speed of the motor, and how it relates to the speed in S.I. (and other) units can be seen in the following section.

5.2 Speed (v) units' conversions

5.2.1 Degrees per second to microsteps per second

$$v_{microsteps/s} = \frac{v_{degrees/s}}{microstep\ angle^{\circ}}$$

Equation 6 Degrees per second to microsteps per second

5.2.2 Radians per second to microsteps per second

$$v_{microsteps/s} = \frac{180 \times v_{radians/s}}{\pi \times \ microstep\ angle^{\circ}}$$

Equation 7 Radians per second to microsteps per second

5.2.3 Revolutions per minute to microsteps per second

$$v_{microsteps/s} = \frac{6 \times v_{revolutions/min}}{microstep\ angle^{\circ}}$$

Equation 8 Revolutions per minute to microsteps per second

5.2.4 Microsteps per second to microstep period (T) in seconds and microseconds

The speed in microsteps per second can be seen as the microstep frequency.

Hence the microstep period (T) is;

$$T_s = \frac{1}{v_{microsteps/s}}$$

Equation 9 Microsteps per second to microstep period in seconds

This value can be used by the microcontroller to control the pulses to the stepper driver by introducing a delay between pulses. This will be required in microseconds;

$$T_{\mu s} = \frac{1000000}{v_{microsteps/s}}$$

Equation 10 Microsteps per second to microstep period in microseconds

6 Acceleration and deceleration

Stepper motors can be driven without acceleration/deceleration. The motor will appear to start/stop (almost) instantly, however, the motor will not be able to produce enough torque to achieve its maximum speed. Also, doing so may result in a large jerk effect upon starting and stopping, especially at the top end of the speeds achievable when the motor is driven this way. In order to provide a "softer" start and stop, and achieve the maximum speed of the motor, acceleration and deceleration should be implemented.

Similar to the microstep period, the concept of the speed period (Y) can be considered. Given that the speed of the motor is controlled by the frequency of the pulses generated, the acceleration can be considered to be controlled by the frequency at which the speed/microstep period is changing. The interval between the change is the speed period.

The concept of the speed period will not be taken further. Reasons for this are discussed in section 6.3.1 where it made sense to implement an alternative time interval in order to simplify the development of an acceleration algorithm. The formula to obtain the speed period given in the following section for completeness. In order to fully implement acceleration and deceleration, speed profiles (section 6.2) as well as acceleration algorithms (Section 6.3) must be considered.

6.1 Acceleration (a) units' conversations

6.1.1 Degrees per second squared to microsteps per second squared

$$a_{microsteps/s^2} = \frac{v_{degrees/s^2}}{microstep\ angle^{\circ}}$$

Equation 11 Degrees per second squared to microsteps per second squared

6.1.2 Radians per second squared to microsteps per second squared

$$a_{microsteps/s^2} = \frac{180 \times a_{radians/s^2}}{\pi \times\ microstep\ angle^{\circ}}$$

Equation 12 Radians per second squared to microsteps per second squared

6.1.3 Revolutions per minute squared to microsteps per second squared

$$a_{microsteps/s^2} = \frac{6 \times a_{revolutions/min^2}}{microstep\ angle^{\circ}}$$

Equation 13 Revolutions per minute squared to microsteps per second squared

6.1.4 Microsteps per second squared to speed period (Y) in seconds and microseconds

The acceleration in microsteps per second squared can be seen as the speed frequency.

Hence the speed period/delay (Y) is;

$$Y_s = \frac{1}{a_{microsteps/s^2}}$$

Equation 14 Microsteps per second squared to speed period in seconds

Theoretically, this value can be used by the microcontroller to control the rate of change of the microstep period by introducing a delay between the change. This will be required in microseconds;

$$Y_{\mu s} = \frac{1000000}{a_{microsteps/s^2}}$$

Equation 15 Microsteps per second squared to speed period in microseconds

As discussed in the previous section, the concept of the speed period will not be taken further.

6.2 Speed profiles

Let;

v_{set} = The set angular speed (microsteps/s).

$v_{achievable}$ = The achievable angular speed (microsteps/s).

a_{set} = The set angular acceleration/deceleration (microsteps/s^2).

t_a = Time (s) to achieve the set speed.

d_a = Minimum distance/swept angle (microsteps) required to accelerate to/decelerate from the set speed.

d_c = Distance/swept angle (microsteps) required for constant speed motion.

D_a = Total minimum distance/swept angle (microsteps) required for acceleration and deceleration.

D = Distance/angle to move (microsteps).

6.2.1.1 Triangular speed profile

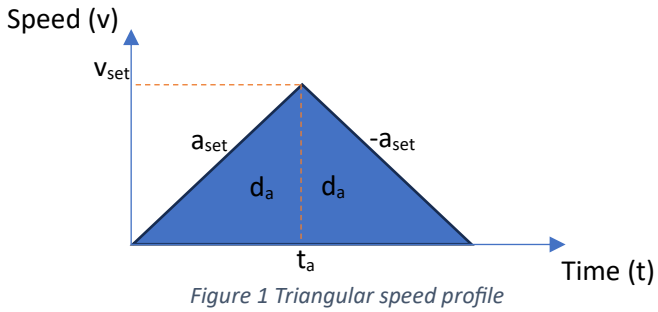

Figure 1 Triangular speed profile

The first and second right-angled triangles in Figure 1 represent the acceleration and deceleration regions respectively.

a_{set} is given by the slope of the hypotenuse, therefore;

$$t_a = \frac{v_{set}}{a_{set}}$$

Equation 16 t_a

d_a is given by the area underneath the right-angled triangle, therefore;

$$d_a = \frac{t_a \times v_{set}}{2} = \frac{(v_{set})^2}{2 \times a_{set}}$$

Equation 17 d_a

$$D_a = 2 \times d_a$$

Equation 18 D_a

Triangular speed profile should be used if $D \leq D_a$

Accelerate until the remaining distance/angle to move is ≤ D / 2

Decelerate until the remaining distance/angle to move is 0.

Note, the achievable speed for each situation is as follows:

If $D = D_a$ the motor will accelerate to;

$$v_{achievable} = v_{set}$$

Equation 19 $v_{achievable}$ (Triangular speed profile when $D = D_a$)

If D < D_a the motor will accelerate to a value which is less than v_{set};

$$v_{achievable} = \sqrt{2a_{set}D}$$

Equation 20 $v_{achievable}$ (Triangular speed profile when D < Da)

6.2.1.2 Trapezoidal speed profile

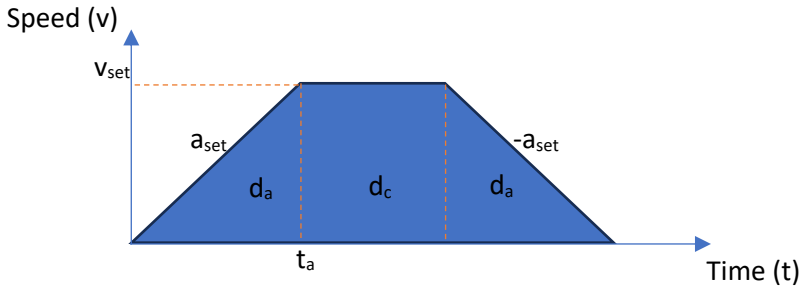

Figure 2 Trapezoidal speed profile

As before, the first and second right-angled triangles in Figure 1 represent the acceleration and deceleration regions respectively. The rectangle represents the constant speed region.

Equation 17 applies for the acceleration and deceleration regions, therefore;

$$d_c = D - D_a = D - (2 \times d_a)$$

Equation 21 d_c

Trapezoidal speed profile should be used if D > D_a

Accelerate until the remaining distance/angle to move is ≤ $d_c + d_a = D - 2d_a + d_a = D - d_a$

Move at constant speed until the remaining distance/angle to move is ≤ d_a

Decelerate until the remaining distance/angle to move is 0.

Hence, the achievable speed ($v_{achievable}$) for the trapezoidal speed profile is always given by Equation 19.

6.2.1.3 Changing speed and acceleration mid-motion

If a change in the value of acceleration occurs during motion, the value of D_a must be recalculated using the new value of a_{set}, and the original value of the travel distance D. Then the value of D_a can be compared with the actual value of D i.e., minus the steps already taken before the value of acceleration changed.

If a change in the value of speed occurs during motion, there are a number of possible situations, however, the simplest approach would be to recalculate for the speed profiles as described for acceleration.

6.3 Acceleration algorithms

6.3.1 v = u + at "simplified"

The following algorithm has been developed and implemented as part of the MT-arduino-stepper-driver library project.

Consider the equation of motion for linear angular acceleration;

$$v = u + a\Delta t$$

Equation 22 v = u + aΔt

Where;

v = Final angular speed (microsteps/s).

u = Initial angular speed (microsteps/s).

a = Angular acceleration (microsteps/s^2).

Δt = Time period (s).

This must be implemented incrementally in order for the microcontroller to accelerate the stepper motor. Hence Δt is also the period of time between calculations.

Let;

i = Current iteration = 1, 2, 3, …

$$\left(v_{microsteps/s}\right)_i = \left(v_{microsteps/s}\right)_{i-1} + a_{microsteps/s^2} \times \left(\Delta t_s\right)_{i-1}$$

Equation 23 $v_{microsteps/s}$ (General iterative form)

When applying this algorithm on a microcontroller, it makes sense to calculate the new speed after each microstep of the motor. This helps simplify the above equation since the period of time between each calculation equals the current microstep period.

Hence, as per Equation 9; $(\Delta t_s)_{i-1} = 1 / (v_{microsteps/s})_{i-1}$ and Equation 23 becomes;

$$\left(v_{microsteps/s}\right)_i = \left(v_{microsteps/s}\right)_{i-1} + \frac{a_{microsteps/s^2}}{\left(v_{microsteps/s}\right)_{i-1}}$$

For i > 1

Equation 24 $v_{microsteps/s}$ (i > 1), accelerating

Applying Equation 24 to Equation 10;

$$\left(T_{\mu s}\right)_i = \frac{1000000}{\left(v_{microsteps/s}\right)_i}$$

Equation 25 $T_{\mu s}$

For the initial value at i = 1, $(\Delta t_s)_0$ can be obtained using;

$$\Delta s = u\Delta t + \frac{1}{2}a\Delta t^2$$

Equation 26 Δs = uΔt + ½ aΔt^2

Where;

Δs = Distance/angle moved (microsteps).

As previously mentioned, the new speed is calculated after each microstep of the stepper motor, therefore, each microstep (including the initial one) is given by Δs = 1. Also, initial speed u = 0, hence, after re-arranging and applying the iterative form, Equation 26 becomes:

$$(\Delta t_s)_0 = \sqrt{\frac{2}{a_{microsteps/s^2}}}$$

Equation 27 Δt_s (i = 1), from stand-still

Applying Equation 27 to Equation 23;

$$\left(v_{microsteps/s}\right)_1 = a_{microsteps/s^2} \times \sqrt{\frac{2}{a_{microsteps/s^2}}}$$

Equation 28 $v_{microsteps/s}$ (i = 1), from stand-still

As before; Equation 25 applies to Equation 28 to obtain $T_{\mu s}$

For deceleration, acceleration is negative, therefore, Equation 24 becomes;

$$\left(v_{microsteps/s}\right)_i = \left(v_{microsteps/s}\right)_{i-1} - \frac{a_{microsteps/s^2}}{\left(v_{microsteps/s}\right)_{i-1}}$$

Equation 29 $v_{microsteps/s}$ (i > 1), decelerating

In summary, apply Equation 28 to obtain the initial values, Equation 24 during the acceleration phase, and Equation 29 during the deceleration phase.

6.3.1.1 The algorithm summarised

This summary simplifies the equations further by introducing the concept of a motion phase multiplier (K) to specify when the motor is accelerating, moving at constant speed or decelerating.

At i = 1, Equation 28 is brought forward as is, to determine the initial velocity;

$$\left(v_{microsteps/s}\right)_1 = a_{microsteps/s^2} \times \sqrt{\frac{2}{a_{microsteps/s^2}}}$$

Equation 30 $v_{microsteps/s}$ (i = 1), from stand-still

For i > 1, the motion phase multiplier K is applied to Equation 24 and Equation 29 to unify them;

$$\left(v_{microsteps/s}\right)_i = \left(v_{microsteps/s}\right)_{i-1} + K\frac{a_{microsteps/s^2}}{\left(v_{microsteps/s}\right)_{i-1}}$$

Equation 31 $v_{microsteps/s}$ (i > 1), accelerating/decelerating

Finally, Equation 25 is brought forward as is, to obtain $T_{\mu s}$ for Equation 30 and Equation 31;

$$(T_{\mu s})_i = \frac{1000000}{(v_{microsteps/s})_i}$$

Equation 32 $T_{\mu s}$

Where;

K = 1 for acceleration,

K = 0 for constant speed,

K = -1 for deceleration.

Sample implementation code can be found at:

https://github.com/jo3-tech/jm-arduino-examples/tree/main/accelerate-stepper-motor-joseph-morgridge-24-algorithm

URL 5 Link to sample arduino code implementing the acceleration algorithm developed during this project

The following sections present algorithms in the literature which are also considered for use in the library.

6.3.2 Algorithm by Austin, 2005

The algorithm (Austin, 2005) can be summarised by the following equations:

$$C_n = C_{n-1} - \frac{2C_{n-1}}{4n+1}$$

Equation 33 C_n

The initial value at n = 0 is given by;

$$C_0 = 0.676f\sqrt{\frac{2}{a}}$$

Equation 34 C_0

Where;

Cn = Microstep period T (µs).

a = Angular acceleration (microsteps/s^2).

n = Iteration count.

f = Timer frequency (Hz) = 1 s / (10^{-6}) s = 10^6 Hz because the smallest timer count is 1 µs (for the Arduino platform).

Sample implementation code can be found at:

https://github.com/jo3-tech/jm-arduino-examples/tree/main/accelerate-stepper-motor-david-austin-05-algorithm

URL 6 Link to sample arduino code implementing the acceleration algorithm by D. Austin, '05

6.3.2.1 AccelStepper stepper motor library

This is a well-known library, created by Mike McCauley, and it implements the algorithm by David Austin.

https://github.com/waspinator/AccelStepper

URL 7 Link to the AccelStepper library repository

Sample code can be found in the library's examples folder.

6.3.3 Algorithm by Eiderman, 2004

The algorithm (Eiderman, 2004) can be summarised by the following equations:

$$p = p(1 + mpp)$$

Equation 35 p

Also;

$$R = \frac{a}{F^2}$$

Equation 36 R

m = -R during acceleration phase

m = 0 between acceleration and deceleration phases

m = R during deceleration phase

The initial value is given by;

$$p_1 = \frac{F}{(v_0{}^2 + 2a)^{1/2}}$$

Equation 37 p_1

Where;

p = Microstep period T (µs).

a = Angular acceleration (microsteps/s^2).

v_0 = Initial/base speed (microsteps/s) = 0 if starting from stand-still.

F = Timer frequency (Hz) = 1 s / (10^{-6}) s = 10^6 Hz because the smallest timer count is 1 µs (for the Arduino platform).

Optional enhancements (trade off (processing) speed to gain accuracy);

$$p = p(1 + q + qq)$$

Equation 38 p for higher accuracy

$$p = p(1 + q + 1.5qq)$$

Equation 39 p for even higher accuracy

Where;

q = mpp

Sample implementation code can be found at:

https://github.com/jo3-tech/jm-arduino-examples/tree/main/accelerate-stepper-motor-aryeh-eiderman-04-algorithm

URL 8 Link to sample arduino code implementing the acceleration algorithm by A. Eiderman, '04

6.3.4 Comparison of acceleration algorithms

The following materials and parameters were used for the comparison;

Stepper motor: Anycubic Nema 17 bipolar stepper motor with 0.4 Nm maximum operating torque.

Stepper driver: Unbranded TB6600 stepper driver.

Microcontroller: Arduino Uno R3.

Parameters are as follows (values highlighted in yellow are manual inputs, others are calculated):

Microstep mode	Step angle (degrees)	Gear ratio	Microstep angle (degrees)	Speed (RPM)	Speed (microsteps/s)	Microstep period (µs)	Acceleration (microsteps/s^2)
16	1.8	1	0.1125	150	8000	125	3000

Table 3 Test parameters for comparison of acceleration algorithms

Sample code at URL 5, URL 6, and URL 8, were used to obtain the following results:

Algorithm	**This project**	**D. Austin '05**	**A. Eiderman '04**
New speed calculation time (µs)	76	76	44

Table 4 Test results: New speed calculation time

The results show that the performance of the algorithms is relatively closely matched with the exception of the algorithm from (Eiderman, 2004), which is almost double the speed of the other two. There was no observable difference in the physical performance of the motor, as each successfully accelerated the motor without stalling. Measuring the accuracy of the algorithms is not trivial, and is beyond the scope of this report.

For completeness, the values of microstep period generated by each algorithm was observed (See the Appendix) for a few iterations and again the results are comparable. The values from each algorithm get more closely matched at higher iteration counts, however only the first 25 iterations are shown.

7 Summary and conclusion

The equations required to implement a stepper motor control library have been presented. An algorithm for acceleration/deceleration was developed and compared with two other algorithms in the literature. It was found that the algorithm developed in this project closely matched the performance of the other algorithms. Taking into consideration that all three algorithms can be made compatible, it was decided to implement each of them as a user-selectable option in the MT-arduino-stepper-driver library.

It should be noted that the performance in software, of the algorithms presented here, is limited by the relevant hardware, in particular, the processing speed of the system (microcontroller and stepper driver), as well as the achievable acceleration and speed of the motor being driven.

8 References

Austin, D. (2005, January). Generate stepper-motor speed profiles in real time. *Embedded Systems Programming*.

Basics of Stepper Motors. (2024, 09 11). Retrieved from Oriental motor: https://www.orientalmotor.com/stepper-motors/technology/stepper-motor-basics.html

Eiderman, A. (2004). *Real Time Stepper Motor Linear Ramping Just by Addition and Multiplication.* Retrieved from HWML: http://hwml.com/LeibRamp.htm

9 Appendix

Acceleration (microsteps/s^2)	i	v (microsteps/s)	T (µs)
3000	0	77.45966692	12910
	1	116.18950039	8607
	2	142.00938936	7042
	3	163.13475307	6130
	4	181.52445816	5509
	5	198.05115672	5049
	6	213.19875822	4690
	7	227.27013474	4400
	8	240.47028532	4159
	9	252.94583919	3953
	10	264.80608568	3776
	11	276.13513045	3621
	12	286.99937650	3484
	13	297.45236088	3362
	14	307.53800952	3252
	15	317.29290130	3152
	16	326.74788749	3060
	17	335.92927814	2977
	18	344.85972927	2900
	19	353.55891837	2828
	20	362.04406708	2762
	21	370.33035119	2700
	22	378.43122651	2642
	23	386.35869070	2588
	24	394.12349598	2537
	25	401.73532332	2489

Table 5 Values of microstep period for the acceleration algorithm from this project

Acceleration (microsteps/s^2)	f	n	Cn (µs)
3000	1000000	0	17454
		1	10473
		2	8145
		3	6892
		4	6081
		5	5502
		6	5062
		7	4713
		8	4427
		9	4188
		10	3984
		11	3807
		12	3651
		13	3513
		14	3390
		15	3279
		16	3178
		17	3086
		18	3001
		19	2924
		20	2851
		21	2784
		22	2722
		23	2663
		24	2608
		25	2557

Table 6 Values of microstep period for the acceleration algorithm from D. Austin '05

Acceleration (microsteps/s^2)	f	R	m	v0 (microsteps/s)	n	P (µs)
3000	1000000	0.000000003	-0.000000003	0	0	12910
					1	6455
					2	5648
					3	5108
					4	4708
					5	4395
					6	4140
					7	3927
					8	3746
					9	3588
					10	3449
					11	3326
					12	3216
					13	3116
					14	3025
					15	2942
					16	2866
					17	2795
					18	2730
					19	2669
					20	2612
					21	2558
					22	2508
					23	2461
					24	2416
					25	2374

Table 7 Values of microstep period for the acceleration algorithm from A. Eiderman '04