Synchronized Control of Four or More Stepper Motors for Computer Numerical Controled Machines and 3D Printers

Nikola Jovanovski[1] and Josif Kjosev[2]

[1,2]*Institute of Electronics, Faculty of Electrical Engineering and Information Technologies, Ss. Cyril and MethodiusUniversity, Skopje, Republic of Macedonia*

jnikola@feit.ukim.edu.mk, josif@feit.ukim.edu.mk

Keywords: CNC machine, CNC machining, CNC milling machine, 3D printer, 3D printing, Stepper motor control.

Abstract: A simple and effective algorithm for synchronized control of four or more stepper motors for computer numerical control machines is presented. Advantages and disadvantages of stepper and servo motors for CNC machines are discussed. Various control topologies for stepper motors are analyzed. Different types of stepper motor drivers and their characteristics are explained. The main emphasis is on the algorithm for synchronous control of multiple motors. The mathematical background for moving functions is explained in context of an overall 3D printing process. A practical implementation of the algorithm is evaluated. Possible directions for further research in this area are pointed.

1 INTRODUCTION

The Computer Numerical Control (CNC) is a technology that uses microcomputers to generate, parse and execute the sequential control that describes the end effector's behavior. The application of this technique is often used in turning, drilling, milling or extruding machines. Recently its application has expanded to other tasks, such as: electronic components insertion, tube welding, and cutting robots [1]. While these technologies are at the top of the third industrial revolution, today we are facing the fourth industrial revolution where one of the main driving forces will be 3D printing [12].

The idea behind this paper is to research the possibilities for designing an electro-mechanical system that can be used both as a 3D printer and a CNC machine. The focus in this paper is placed on the 3D printer issues, specifically, the synchronized control of 4 or more stepper motors. These results can be reused for CNC milling machines with some additional components [5].

The starting points are: the mechanical construction, mechanisms for movement and their integration. A 3D model of the machine is shown in figure 1. Speed and forces (torques) calculations define the motors and motor drivers selection. There are various types of printer heads but for this dual-purpose machine a thermal extruder type appears to be the most appropriate. The main controller should be also suitably selected for the dual purpose. These issues have been discussed in a previous paper [5].

CNC machine motion can be performed with AC or DC servo motors or stepper motors. Servo motors require additional positioning sensors and closed loop control systems. Stepper motors are very accurate in open loop systems [2][5] and are simple to control provided that the maximal required torque at the maximal required speed is available [4][6].

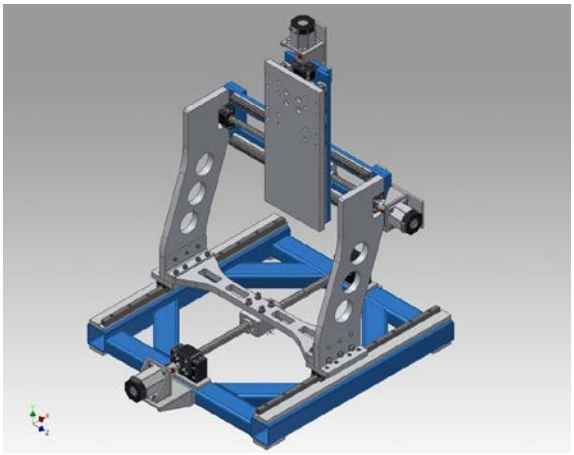

Figure 1: 3D printer model.

A specific issue regarding motion control is the multiple axes motion synchronization. This is a very complex task with servo motors but is also nontrivial with stepper motors. 3D printer manufacturers have proprietary control algorithms that are not accessible for performing experiments. Open source community projects are available for this purpose but, again, better performing algorithms are usually encapsulated in binary libraries. Some motor and controller manufacturers offer PLC controllers that provide 2-axes synchronization with speed control [13] but PLCs are an expensive overkill with the additional (unnecessary) options they have. They also do not provide solutions for 3 or more axis synchronization. In this paper we present a very efficient algorithm for synchronized stepper motors control that uses small number of operations compared with the algorithm for linear interpolation explained in the literature [3]. The results are based on real experiments with a 3D printer designed from scratch in our laboratory to experiment with different types of stepper motors and algorithms.

2 STEPPER MOTOR DRIVE TOPOLOGIES

The following is a brief review of the stepper motor operation and their drive topologies. Stepper motors usually have permanent magnet rotor and two (or four) groups of windings on the stator as presented in figure 2. The group of windings is presented as coils at 90 degrees which are called “electrical degrees” while the windings are positioned at certain spatial angle such that a 360° electrical angle corresponds to a much smaller spatial angle (eg. 7.2°).

The rotor is moved by changing the current through the coils in appropriate sequence such that magnetic forces attract or repel the rotor to support the rotation. If the currents through the coils change direction, the motor is named bipolar. Otherwise it is unipolar.

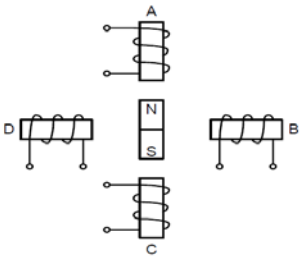

Figure 2: Stepper motor principle diagram.

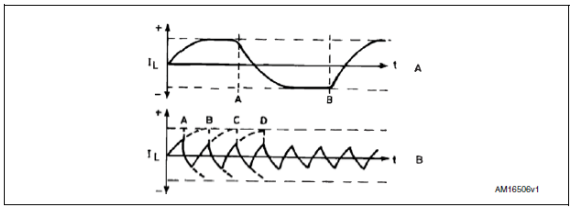

Figure 3: Peak winding current decreasing at higher speed, limited by motor inductance [9].

The current is obtained by connecting the coils to a voltage source V through appropriate electronic switches (a stepper motor driver). One step is a sequence of two coils (or four coils) sequentially energized/de-energized (pulsed) and it corresponds to a rotation by a predefined angle (typically 1.8° for a 200-step motor).

The coils resemble first order circuits with their inductance L and resistance R. This means the currents through the coils change exponentially:

$$i(t) = I(\infty) - \left[I(\infty) - I(0) \right] e^{-t/\tau}$$

where $I(0)$ is the starting current, $I(\infty)=V/R$ is the steady state current and $\tau=L/R$ is the time-constant of the circuit. A complete current change takes approximately 3τ. This limits the maximal applicable step rate i.e. rotational speed of the motor at full torque because the torque depends on the coil current amplitude – figure 3.

The time-constant can be reduced by increasing R, and correspondingly V, externally by a same factor n (typically 4-5) to retain the nominal current. Unfortunately this increases the power loses nRI^2 to an unacceptable level. The solution is the Pulse Width Modulation (PWM) technique which can provide (on average) the nominal coil current at much higher voltage with minimum power loses – figures 4 and 5.

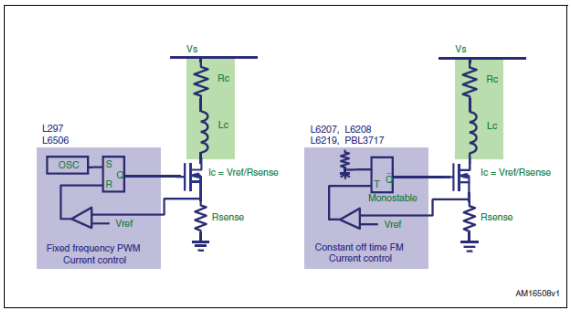

Figure 4: Switch mode drive implementation [9].

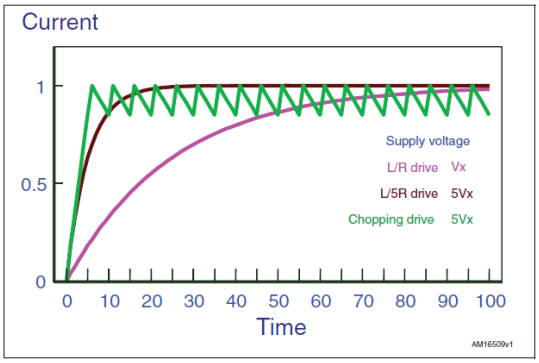

Figure 5: Current waveforms for L/R, L/nR and Switch mode drive [9].

3 STEPPER MOTOR DRIVERS

Simplest stepper motor drivers provide full-step and sometimes half-step driving. For a two-phase motor a full step consists of complete sequential de-energizing and energizing of the two coils and provides rotation of 90° electrical. Half-step is achieved by energizing both coils with the same polarity. This positions the rotor at 45° electrical and doubles the number of pulses for a revolution but increases the positioning resolution.

The previous idea can be extended further: if the currents in the energized coils are not equal, the rotor can be positioned at any angle between 0 an 90°, depending on the current ratio. This technique is called microstepping and provides substantial increase of the positioning resolution. Modern PWM stepper controllers are ideal for providing the microstepping control technique. They can typically increase the number of steps up to 16 times (eg. HY-DIV268N) or up to 256 times (eg. DQ860MA).

CNC machines convert the motor rotation into linear motion by ball screws. A standard value of 5mm displacement is obtained for each motor revolution. Thus a typical 200-pulse motor provides 0.025mm linear resolution. This is much above the required resolution. If microstepping with 16 microsteps is applied, the resolution becomes 1.56 micrometer which is satisfactory [11]. A 256 microsteps would provide resolution below 0.1 micrometer that is more than satisfactory [10]. Actually, uncompensated motor errors are much higher than this resolution [14] when absolute positioning is needed but position calculations can also accumulate errors that should be kept below the CAD resolution (which is typically at 1 micrometer). Additional benefits of high frequency microstepping are smooth motion, reduced motor noise, and reduced possibility for electro-mechanical resonance and oscillations [14].

Typical stepper motor driver has three control inputs:

- ENABLE/DISABLE (“zero” means the driver is disabled and the motor is turned off, while “one” puts the motor in the hold torque state).
- DIRECTION (“zero” drives the motor in clockwise direction, and “one” drives the motor in counter-clockwise direction)
- PULSE (each pulse at this input means rotation of 1 (micro-)step of the motor in the direction selected by DIRECTION input)

These signals are generated by the control unit’s firmware. Central part of this firmware is the optimized software algorithm for multiple motor driving, which is presented in the following sections.

4 3D PRINTER PRACTICAL IMPLEMENTATION

Besides the previously mentioned electromechanical components, the complete 3D printer includes:

-CAD software for designing and rasterizing the parts to be printed and generating commands in the standard CNC language named G-code,

-control unit (microcontroller),

-methods for transferring the G-code to the control unit,

-control unit firmware for coordinates calculations and motor driving signals generation,

-user interface.

The implemented CAD software is Pronterface (free open-source software) with it’s rendering engine Slic3r that slices the part into printable layers. The generated G-code specifies sequence of coordinates for linear X-Y axes movement and the required speed for each movement including the extruder for material deposition as coordinate E. The movement in Z direction is generated only once for each layer to be printed.

The control unit is based on STM32F4Discovery board with a CortexM4 microcontroller running on 168MHz.

G-commands are transferred through serial communication at the rate of 115 kbps. New command is transferred after the current is completed.

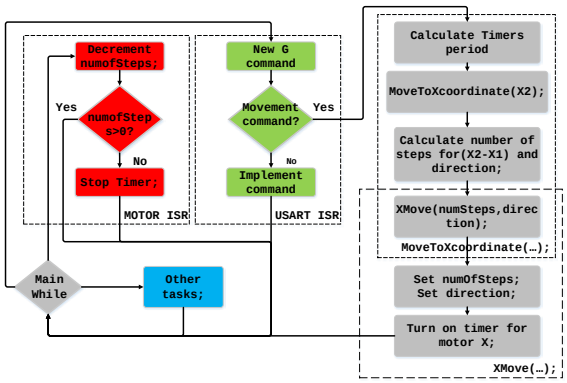

Figure 6: Firmware principle block diagram.

The firmware of the controller consists of many functions (communication, coordinate interpolation, motor control, extruder heating PID control, sensor scan, user interface through a graphical touch-screen, and some other housekeeping functions). Motor control functions are timer interrupt-driven and have the highest priority. Communication is also interrupt driven at a lower level and incorporates the linear interpolation calculations. The lowest interrupt driven function is the PID heater control (not shown here). All other functions are reaized in an infinite loop. The principal block-diagram of the firmware is presented in figure 6.

The current position is tracked through four global variables (*X*,*Y*,*Z*,*E*) which are updated on each motor step. The target position, sent by the CAD system, is kept in the variables *X*2, *Y*2, *E*2 and *speed*, (As previously mentioned, Z movement is accomplished separately between layers and will not be discussed here.)

The origin (0,0,0,0) is established at power-up of the machine. The printer moves into negative direction until proximity switches are activated to set the spatial origin (0,0,0) while the extruder coordinate at start-up is 0 by definition.

5 SOFTWARE IMPLEMENTATION OF LINEAR INTERPOLATION BETWEEN TWO POINTS

The need for linear interpolation can be depicted in figure 7. If movement is to be accomplished from point (X1,Y1) to point (X2,Y2) and the required numbers of steps for the X and Y axes movements are generated with the same frequency, a movement along the dotted line will be performed instead of a movement along the dashed line.

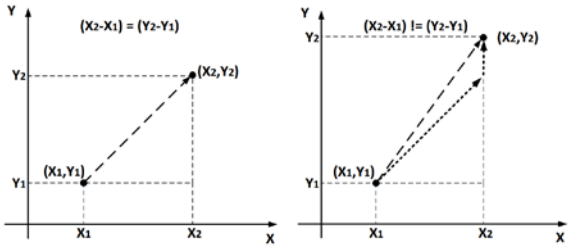

Figure 7: Movement path between two points without linear interpolation.

Obviously, the speeds along X and Y axes must be different to accomplish the required movement along the dashed line. The distance and the speed components along the X and Y axis are shown in Figure 8.

The problem here is to determine the values of the X and Y components of the velocity. Since the CAD system actually defines the speed of the material extrusion *V* (mm/min) it can be applied only to the direct distance between the points (1) and (2), Therefore first we needed to calculate the distance *S* from the starting point to the destination point. We calculate the coordinate changes ΔX and ΔY from the following equations:

$$\Delta X = | X_2 - X_1 |, \Delta Y = | Y_2 - Y_1 | .$$

and then the distance :

$$S = \sqrt{\Delta X^2 + \Delta Y^2}$$

Now the travel time is determined from the following equation:

$$t = \frac{S[mm]}{V[mm/\min]}.$$

Since the extruder travels the distances ΔX and ΔY during time t, the components of the velocity can be calculated by the following equations:

$$V_X = \frac{\Delta X[mm]}{t[\min]}, V_Y = \frac{\Delta Y[mm]}{t[\min]}.$$

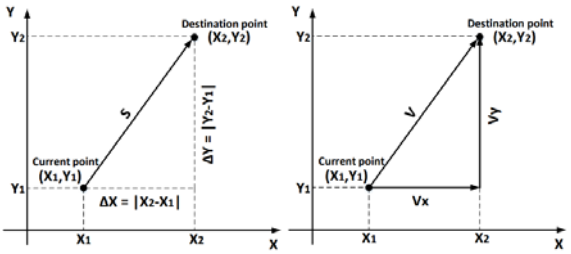

Figure 8: Interpolation of distance and velocity between two points.

The number of pulses for the required movement in the X and Y directions, ΔX and ΔY, can be calculated from the known koeficient of the spindle, 5mm/revolution, number of steps per revolution (200) and number of microsteps per step of the motor controler. Similar calculation holds for the extruder but here the distance is S.

The velocity in each direction is determined by the frequency of steps the motor performs i.e. the frequency of pulses applied to the PULSE imput of the controller. The easiest way to generate pulses with certain frequency (or period) is to use a timer/counter in PWM mode and set the period of the PWM waveform. The duty cycle is not important because the motor controller takes care of the PWM duty cycle applied to the motor. Therefore the pulse duration can be set to the maximum applicable value. If the current in the coil rampes above the maximal value, it would be limited by the power supply current limit.

By seting the timer prescalers, depending on the timer frequency and drivers microstepping factors, the following calculation can be established:

$$T_X = \frac{CONSTx}{V_X[mm/\min]}[us], T_Y = \frac{CONSTy}{V_Y[mm/\min]}[us].$$

Similar calculation holds for the extruder:

$$T_E = \frac{CONSTe}{\left|E_2 - E_1\right|} \cdot t[us]$$

where CONSTe is the appropriate scaling constant.

These values (T_X, T_Y, T_E) are directly written into the registers that define the timer periods associated with X, Y and E axes which is acomplished with the function:

```
void MoveXY(double x2, double y2,
double e2, double speed);
```

The same function calculates the number of steps for each coordinate and activates the appropriate timer. This is presented on the right-hand side of figure 6.

6 SOFTWARE IMPLEMENTATION TIMING ANALYSIS

As mentioned before, the current position of the printer is tracked through four global variables (X,Y,Z,E). These variables are updated on every motor step i.e. at every timer pulse in an interrupt routine. (Timers generate interrupts on the falling edge of every pulse). The interrupt routine (only for X-axis) is presented in red colour in Figure 6.

After completing the current movement, new point is obtained, calculations are performed and the process is repeated until a stop command is received.

The firmware is heavily interrupt driven and timing calculations have to be performed to check the processor utilization and confirm that every task can be completed.

The realized prototype has maximum linear speed of 600mm/min, microstepping factor of 128, 200 steps per motor revolution and 5mm linear motion per revolution. This gives the lowest interrupt period of T_m=19.5us. The non-optimized code produced interrupt routines of maximum T_{pm}=0.325us.

The communication between the controller and Pronterface transfers 34 bytes for each point. Since the maximal point distance is 0.1mm the 600mm/min requires 100 transfers per second. The four time_and_number_of_pulses calculation routines take approx. T_{pcalc}=0.35us each and are called at period of T_{calc}=10000us. Each byte is also collected by an interrupt that takes less than T_{pch}=0.15us at a period of approx. T_{ch}=100us.

The total utilization of the processor by the interrupt routines is:

$$Utilization = 4\left(\frac{T_{Pch}}{T_{ch}} + \frac{T_{Pcalc}}{T_{calc}} + \frac{T_{Pm}}{T_m}\right) = 0.073$$

This means that the processor has very low utilization of 7.3% and can easily control much more than 4 motors. This is accomplished by efficient implementation of the motor control algorithm using timer peripherals.

This also leaves space for linear velocity increase up to the maximally obtainable form Pronterface of 2000mm/min and it would not reduce substantially the number of motors because at higher speeds the microstepping factor is also reduced.

The higher speed may interfere with the current baud rate of 115kbps. At 10 bits per byte (1start+8data+1stop) 34 bytes take 3.4ms while the maximal point rate of 20000points/min means 3ms for a point. The remedy would be to increase the baud rate to the maximum specified by Pronterface of 250kbps [15].

7 CONCLUSIONS

This paper presents an efficient and simple algorithm for synchronized control of four or more stepper motors. The algorithm and the 3D printer prototype provide a solid base for further research in this area.

The prototype achieves high resolution at the expense of slightly reduced printing speed. This is due to the massive mechanical construction that provides higher accuracy but also requires higher power drive unit compared to typical commercial printers. On the other hand, the massive construction is needed for the alternative CNC subtractive processing.

Other possible direction for experimenting is variable printing speed. This option is not available in today's commercial 3D printers. Additionally, it should be emphasized that this kind of stepper motor implementation provides a reliable and accurate way of 3D printing without using a feedback link from the motors. This topics will be presented in a following paper.

REFERENCES

[1] Paulo Augusto Sherring da Rocha Junior,Rogerio Diogne de Silva e Souza, Maria Emilia de Lima Tostes, "Prototype CNC Machine Design", 10.1109/INDUSCON.2010.5740068.

[2] J.M. Rosario, L.K. Rincon, D.Kubiak, D. Dumur, L.F.Melo, "Actuator Selection of CNC Machine Tool Based in Dynamical Modeling and Control", International Symposium on Power Electronics, Electrical Drives, Automation and Motion, 2014.

[3] Dev P. Desai, Dr. D.M. Patel, "Design of Control Unit for CNC Machine Tool using Arduino based Embedded System", International Conference on Smart Techologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM), India, 6-8 May 2015, pp. 443-448.

[4] Industrial Circuits Application Note Stepper Motor Basics, http://www.solarbotics.net/library/pdflib/pdf/motorbas.pdf.

[5] Nikola Jovanovski, Josif Kjosev, "Hardware and Software aspects of 3D printers", International Conference ETAI, 2016.

[6] Suk-Hwan Suh, Seong-Kyoon Kang, Dae-Hyuk Chung, Ian Stroud, "Theory and design of CNC Systems, 2008 Springer-Verlag London Limited.

[7] Jacob Bayless, Mo Chen, Bing Dai, "Wire Embedding 3D Printer", Engineering Physics, University of British Columbia, April 12, 2010.

[8] Jonathan Heathcote, "Improving the Makerbot 3D Printer", University of Manchester, School of Computer Science, Project Report 2012.

[9] Thomas Hopkins, Aplication note, Stepper motor driving, http://www.st.com/content/ccc/resource/technical/document/application_note/57/c8/7c/c1/0d/91/46/89/CD00003774.pdf/files/CD00003774.pdf/jcr:content/translations/en.CD00003774.pdf.

[10] Wantai Motor, DQ860MA, PWM currrent control, https://factorydaily.com/fdattachs/fdattachs7/112716014424312.pdf.

[11] HY-DIV268N-5A two phase hybrid stepper motor drive manual, http://wiki.kreitek.org/media/proyectos:cnc:div268n-5a-datasheet.pdf.

[12] Executive Perspectives "Accelerating the Next Industrial Revolution", https://spare-parts-3d.com/2017/04/18/3d-printing-key-driver-4th-industrial-revolution-hp-ceo/

[13] Eaton Logic Controller Programming Manual, http://www.eaton.com/ecm/groups/public/@pub/@electrical/documents/content/mn05003003e.pdf

[14] Ericsson, Industrial Circuits Application Note, Microstepping, http://users.ece.utexas.edu/~valvano/Datasheets/StepperMicrostep.pdf

[15] Plastic Scribbler, Printing the Future in 3D (Pronterface tutorial), http://www.plasticscribbler.com/tutorial/getting-started/item/21-getting-started-with-pronterface#.WlGh4oWcEwE