LUMS Stepper 2.0 – universal 3-axis stepper motor controller

Kacper Oreszczuk

1 Motivation

Stepper motors are commonly used in laboratory. They are powering x-y-z translation stages, tunable filters, polarisation optics and many other devices. The goal of this project is to replace commercially available drivers with a cheaper, more reliable and more universal solution.

Cost:

Commercially available stepper motor drivers designed for laboratory use can reach prices of a few hundred US Dollars for a single axis. Proposed solution costs about $40 per axis with small labour cost.

Quality:

Personal experience in our laboratory revealed problems with microstep precision in drivers supplied by one of the leading laboratory equipment manufacturers. I have tested thoroughly stepper motor chip before choosing it for this project.

Features:

Custom solution allowed me to implement important additional features. The most important is automatic compensation of mechanical hysteresis.

2 Hardware design

Single device is able to control three bipolar stepper motors with 1/64 microsteps precision. Maximum current is about $2.0\,\mathrm{A}$ per motor and the supply voltage must be between $7\,\mathrm{V}$ and $24\,\mathrm{V}$. Low level motor driving operation is performed by *TMC2226* driver on *Stepstick* breakout board. Three such drivers are connected to *NUCLEO144* board with *STM32H723ZG* microcontroller. Nucleo board connected via on-board *FT232RL* USB-UART controller to computer provides virtual serial port for communication and power for logic layer of device. Separate power connector is provided on main PCB for motor operation and six high-density D-SUB15 female connectors for motors and encoders. Full schematics of the device is presented in Figure 2. PCB board is double-sided. It is advised to order PCB with SMD assembly. See Figure 3 for PCB layout. The driver is confined in 3D-printed case (Figure 1) protecting it and allowing mounting to optic table. Design files for PCB and case along with source code for microcontroller are available at project page: `github.com/kacperoreszczuk/stepper2/`.

3 Software and protocol

Driver has two basic operation modes - movement with given velocity or movement to given position. In either of the cases the movement of motor is smoothed at starting and stopping. For user convenience speeds and positions are expressed in real units (for example millimeters in case of linear travel stage), not motor steps. End switches (limit switches) of different types are supported by this driver. They provide additional safety, as the driver will stop at the edge of the movement range of, for example, linear translation stage. Limit switches are also used to automatically calibrate absolute position of the stage.

When the mechanical setup of driven actuator is not completely rigid, the mechanical hysteresis can occur. Due to this effect real position of the actuator depends on the direction from which set position is approached. In this project user can specify the amplitude od mechanical hysteresis. Algorithm will track previous movements and compensate the offsets in a manner transparent for the user.

User can communicate with LUMS Stepper via virtual serial port provided by Nucleo board. Protocol of communication is based on single line plain-text commands initiated from computer and on single line reply from driver. For protocol description please refer to Table 1. Full list of accepted commands is presented in Table 2

Command:	ID header value \r
ID	Single digit number – number of motor, counted from 0. Defaults to 0 if not specified.
header	Two lower case characters specifying command type (see table 2).
value	Numerical argument (optional depending on command type). Defaults to 0 if not specified.
\r or \n	Any endline character. Other whitespace before, after or between command parts is ignored.
Reply:	header value \r
header	Driver returns header after every command
value	Optional depending on command type
Error reply:	?\r

Table 1: *Protocol of communication between PC and LUMS Stepper 2.0.*

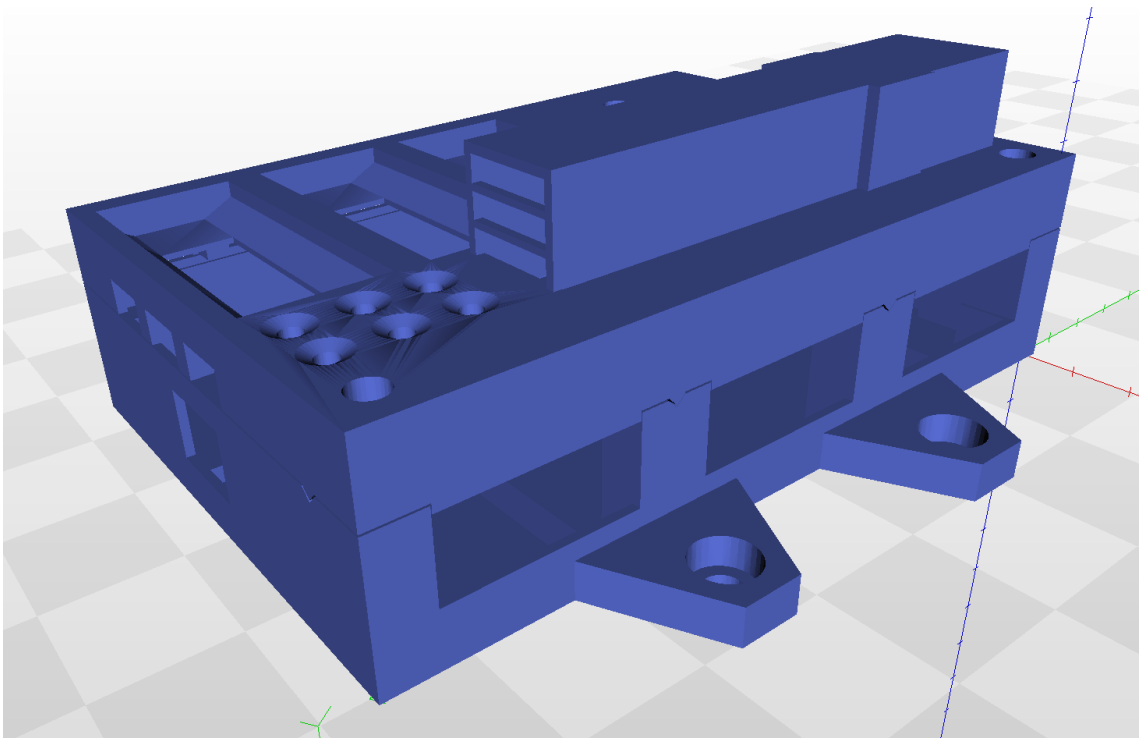

Figure 1: *Case, model for 3D printer.*

Command	Abbreviation origin	Example	Description
id	ID	id	Returns unique ID of device, between 101 and 199.
ac	Axis Count	ac	Returns number of supported motors.
sc	Set Current	0sc600	Sets motor current, in milliamperes.
ss	Set Step	0ss0.005	Defines the size of motor full step in user units, for example in millimetres or degrees.
sh	Set Hysteresis	0sh0.001	Sets mechanical hysteresis compensation distance. Algorithm applies corrections to target positions when necessary (f. eg. when changing direction).
sm	Set Max Velocity	0sm1.5	Sets maximum allowed velocity in move velocity mode, in user units.
sv	Set Velocity	0sv1	Sets velocity in move absolute mode, in user units.
sa	Set Acceleration	0sa0.5	Sets time that it takes to accelerate from zero to move absolute velocity.
sl	Set Limit	0sl2	Sets limit switch type. 0 – no limit switch, 1 - active limit switch (active high). 2/3 – mechanical, default NC/C. 4/5 similarly to 2/3, but enabled only for homing.
sr	Set Reversed	0sr1	Reverses direction of motor rotation and swaps limit switches.
so	Set Offset	0so4.2	Specifies the true position of the limit switch in user units.
hr	Homing Reversed	0hm1	Specifies if homing should be performed to rear limit switch (0, default) or to front limit switch (1).
hm	HoMe	0hm	Starts homing procedure. If limit type is set to zero, then it only corrects current position to be zero.
mv	Move Velocity	0mv0.5	Sets target velocity. Driver will smoothly increase or decrease speed to given value.
ma	Move Absolute	0ma3.2	Sets target position. Driver will smoothly start and halt motor to achieve goal position.
mr	Move Relative	0mr0.1	Changes target position by given value, even if target position was not achieved yet. In „move velocity" mode sets target relative to current position.
tp	Tell Position	0tp	Returns current position of given axis.
ts	Tell Status	0ts	Returns status of current axis. 0 – stopped, 1 – move velocity, 2 - move position, 3 – homing.
ta	Tell All	ta	Returns current positions of all axes, separated by spaces. Next, after another space, returns statuses of all axes in single joined string.
er	Encoder Raw	0er	Returns encoder position in raw counts.
es	Encoder Step	0es0.002	Configures encoder step in user units. Must be called after configuring motor step.
ep	Encoder Position	0ep	Returns encoder position in user units.
ds	Driver Status	0ds	Returns the TMC's DRV$_STATUS register in hex format. See TMC2226 datasheet
dr	Driver Reset	0dr	Cycles the EN pin of the TMC driver, clearing the hardware error flags in the process.
tt	Tell Time	tt	Returns time from boot in seconds.

Table 2: *Commands in communication protocol.*

4 How to set-up Stepper 2.0 in pyLUMS software

This section is very specific to LUMS laboratory.

In order to control Stepper 2.0 from pyLUMS software, you need to configure it in `local_devices.ini` and `devices.ini` file. Most of the work is put in the former one. Here is example of the relevant entry in `local_devices.ini` file:

```
[stepperdevice]
name = My Stepper
class = misc.stepper.StepperWorker
req_port = 7040
pub_port = 7041
id = LUMS01
current = 800,800,800
step = 0.005,0.005,0.005
encoder_step = 0.0001,0.0001,0.0001
```

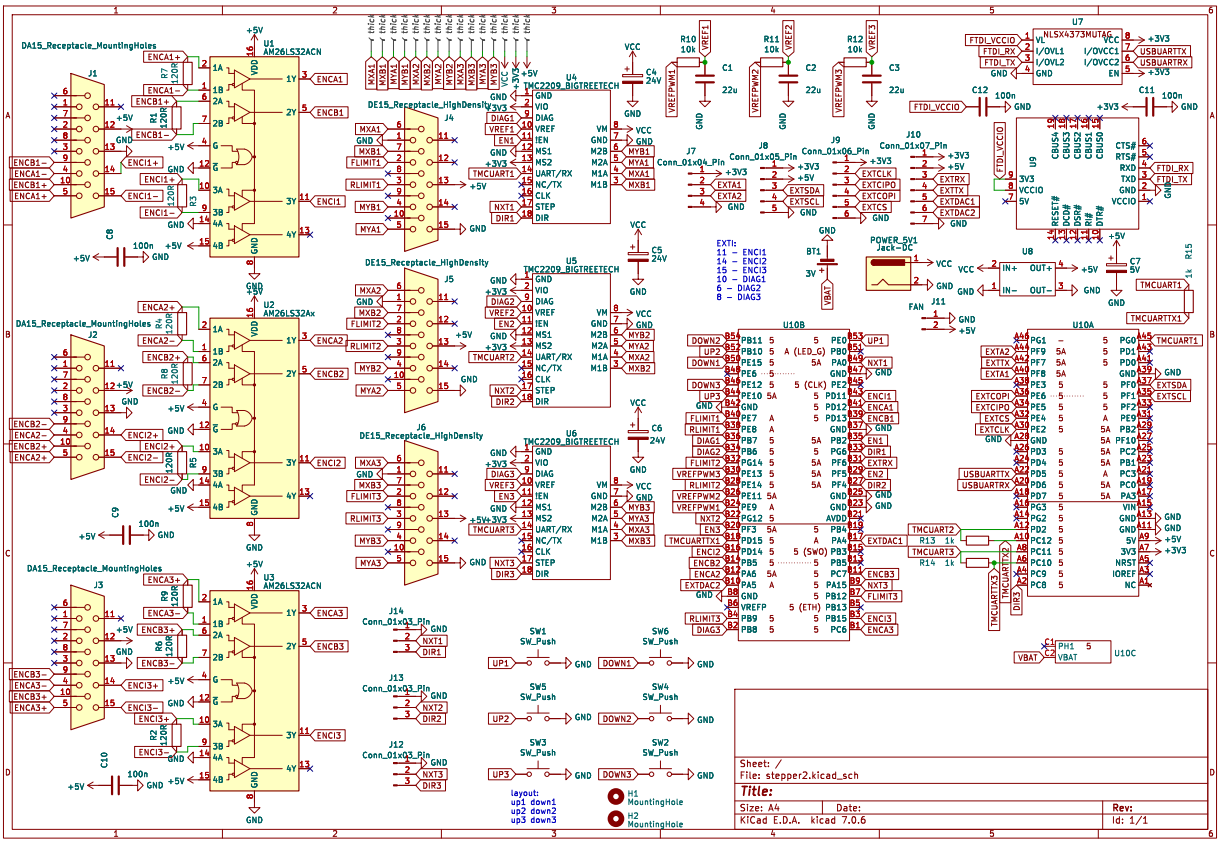

Figure 2: *Schematics of the LUMS stepper.*

```
velocity = 3,3,3
max_velocity = 3,3,3
hysteresis = 0.0012,0.0012,0.0012
homing_offset = 0,0,0
limit_type = 3,3,3
reversed = 0,1,1
acceleration_time = 0.20,0.20,0.20
```

The meaning of each parameter is described in the table 3.

Parameter:	Description
id	The id of the given driver.
current	(for each axis) Move current (in mA) At rest the current is automatically reduced.
step	(for each axis) The distance in the user units corresponding to a single motor step. It does not directly affect the driver, only the values presented to the user.
encoder_step	(optional, for each axis) Distance between encoder marks in the user units. It is relevant only for the stages fitted with the encoder.
velocity	(for each axis) Default move velocity in the user units per second. Note the user can request moving with different velocity.
max_velocity	(for each axis) Maximum move velocity in user units per second for each axis. The software will prohibit the user to go beyond this value.
hysteresis	(optional, for each axis) ...
homing_offset	(optional, for each axis) ...
limit_type	(optional, for each axis) Definition of the limit switches. Possible options: 0 – no limit switches, ...
acceleration_time	(optional, for each axis) Time in seconds to smoothly reach the moving velocity.

Table 3: *Parameters specified in `local_devices.ini` file if you use pyLUMS software*

Compared to multiple parameters defined above, the configuration in `devices.ini` file is relatively simple. Basically you need to match the number of the ports, e.g.:

```
[stepper]
class = misc.stepper.Stepper
req_port = 7040
pub_port = 7041
```

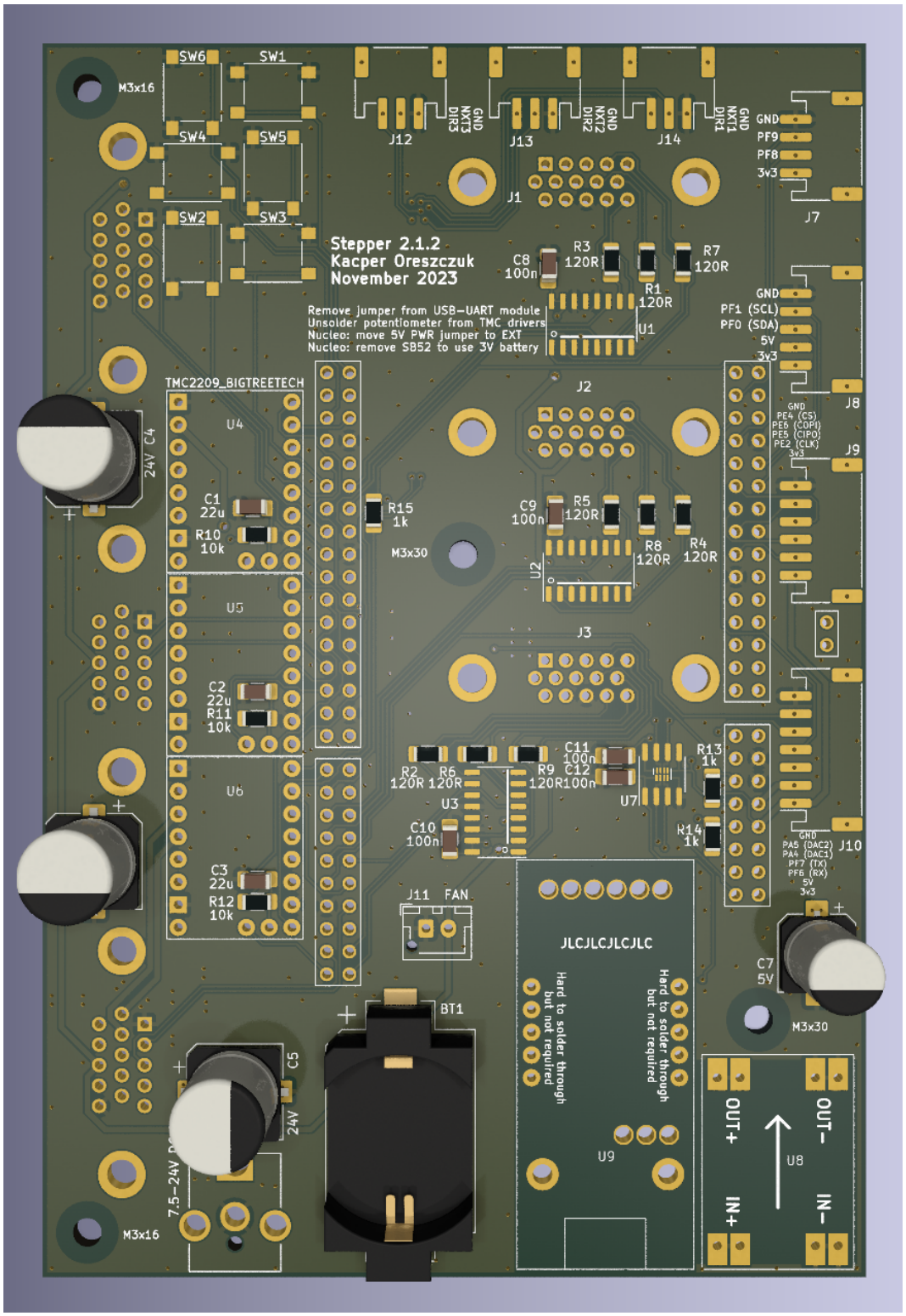

Figure 3: *PCB design. View from top.*