Hangprinter with Marlin (not used)

Marlin Firmware for Hangprinter

> Torbjørn also forked Marlin Firmware and pulled back a lot of stuff into official Marlin feature tree

- https://github.com/MarlinFirmware/Marlin/pull/9180
- github.com/tobbelobb/hangprinter/blob/73a0cf91bbc04a59faccd6f042d30d5d22d170a9/firmware/Marlin/Marlin/Marlin_main.cpp#L819

New gcodes:

- G6
- G95 (torque mode) - G95 Set servo torque mode status. Accepts 0 or 1
- G96 (zero encoders) - G96 Tell sensor servo to mark its reference point
- G97 (measure) - G97 Get sensor servo length travelled since last G96

Adapted gcodes that still use XYZE parameters:

- G0/G1 (inverse kinematics)
- G92

Adapted gcodes that use ABCDE parameters:

- M92
- M201
- M204
- M205
- M906

Adapted gcodes with other parameters

- M665 (set Hangprinter anchor positions / Auto Calibration Simulation Input Parameters)
- M114 S1 (read encoder mm movement since reference point was marked)

New features that also work on non-Hangprinter machines:

- `UNREGISTERED_MOVE_SUPPORT`
- `MECHADUINO_I2C_COMMANDS`

Features that are Hangprinter specific:

- `LINE_BUILDUP_COMPENSATION_FEATURE`

Other changes:

- Uses `NUM_AXIS[_N]` and `MOV_AXIS` instead of `XYZE[_N]` and `XYZ` for indexing and initializing arrays that refer to kinematic axes (movement axes on physical machine) rather than the cartesian axes of incoming gcode.
- Uses `E_CART` instead of `E_AXIS` for indexing arrays that refer to cartesian axes of incoming gcode.

```
G95 A10                 ; Sets the A-motor in torque mode applying 10 units of torque
G95 B15 C15 D20         ; Sets B, C, and D-motors in torque mode
G95 A-35 B-35 C-35 D-35 ; torque mode lower force
G95 A-50 B-50 C-50 D-50 ; torque mode lower force
G95 A0 B0 C0 D0         ; Sets ABCD motors back into position mode


G96 A B C D             ; tells Hangprinter to zero its encoder data on the A, B, C, and D axes.

G97 A B C D             ; asks Hangprinter how many millimeters it has travelled since the previ
G97 A B C D             ; Doing again gives precise numbers about line lenght changes, namely th

G98 A B C D             ; saves the same values into the internal line_lengths variable in firmw
```

Auto Calibration Simulation for Hangprinter (Python script)

The Hangprinter Project has a goal of auto-calibration. That requires locating anchor points by sampling relative line lengths with tight lines at unknown positions. This code tries to optimize the anchor positions to fit the samples of relative line lengths. Note that this code assumes that the B-anchor has a positive x-coordinate. If your machine's B-anchor is negative then your output anchors Bx and Cx will have the wrong signs.

The script will only work if you have closed loop steppers like Smart Stepper or Mechaduino installed. Only with I2C connected hardware a positional feedback is possible to get with M114 S1 command.

Installation on Windows 10 (64 Bit) with Python 3.7.5

Download https://gitlab.com/tobben/auto-calibration-simulation-for-hangprinter

```
pip install mystic scipy numpy
cd C:\auto-calibration-simulation-for-hangprinter>
```

Installation on raspbian (Python 2.7)

> this does not work with older Python 3.5 which is on Raspbian Stretch (my case), so use old Python 2.7.

```
sudo apt-get install libatlas-base-dev python-scipy
pip2.7 install --user mystic # might take half an our to install ...

#get scipy by ready to use wheel file from https://www.piwheels.org/simple/scipy/ because it does not easily compile on a Raspberry Pi (scipy does not compile on raspi https://raspberrypi.stackexchange.com/questions/8308/how-to-install-latest-scipy-version-on-raspberry-pi)
wget https://www.piwheels.org/simple/scipy/scipy-1.3.3-cp35-cp35m-linux_armv7l.whl#sha256=da7755a09413176d91b50de8ce153721927b3e284928d6d86e9b2b1911bb1a69
pip3 install scipy-1.3.3-cp35-cp35m-linux_armv7l.whl

cd /opt
git clone https://gitlab.com/tobben/auto-calibration-simulation-for-hangprinter.git
cd auto-calibration-simulation-for-hangprinter/
```

Test example (run the script)

```
python ./simulation.py

#Output
samples:          11
input xyz coords: 0
total cost:       0.271115
```

```
cost per sample:  0.024647

Warning: Data set might be too small.
         The below values are unreliable unless input data is extremely accurate.

#define ANCHOR_A_Y -1160
#define ANCHOR_A_Z  -141
#define ANCHOR_B_X  1004
#define ANCHOR_B_Y   588
#define ANCHOR_B_Z  -119
#define ANCHOR_C_X  -971
#define ANCHOR_C_Y   518
#define ANCHOR_C_Z  -103
#define ANCHOR_D_Z  2892

M665 W-1160.35 E-141.49 R1003.98 T588.05 Y-118.60 U-971.42 I517.82 O-102.67 P2891.90
```

How to Collect Data Points?

Data collection depends on Smart Stepper and well calibrated line buildup compensation.

- Go into torque mode on all motors and adjust torque magnitude as you prefer.
- Drag mover to the origin and zero counters: `G92 X0 Y0 Z0`
- Mark reference point for all encoders: `G96 A B C D` (Stock Marlin accepts `G96` as a short hand for `G96 A B C D`)
- Repeat 13 - ca 20 times:
  - Drag mover to position of data point collection.
  - Collect data point: `M114 S1` (M114: Get Current Position)

How to Insert Data Points?

Before you run the simulation, open `simulation.py` and modify the main function, near the bottom of the file. Replace `??` with data points collected with your Hangprinter.

```
    ...
    # Replace this with your collected data
    samp = np.array([
[??, ??, ??, ??],
[??, ??, ??, ??]
```

```
        ])
    ...
```

When values are inserted, run again with

```
python ./simulation.py
```

Output Explanation

The first block give some stats trying to describe the quality of the parameters that were found

```
samples:         11
total cost:      0.254896
cost per sample: 0.023172
```

It's recommended to use 13 samples or more. Using fewer samples makes it probable that the solver finds bogus anchor positions that still minimizes cost.

Ideal data points collected on an ideal machine would give `total cost: 0.000000` for any sample size above 10. In real life this does not happen. The `cost per sample` value let you compare results from your different data sets of unequal size.

The second block contains the anchor positions that the script found. They are formatted so they can be pasted directly into Marlin's `Configuration.h`.

```
#define ANCHOR_A_Y -1164
#define ANCHOR_A_Z  -144
#define ANCHOR_B_X   999
#define ANCHOR_B_Y   585
#define ANCHOR_B_Z  -115
#define ANCHOR_C_X  -977
#define ANCHOR_C_Y   519
#define ANCHOR_C_Z  -106
#define ANCHOR_D_Z  2875
```

The gcode line that is the third block can be used to set anchor calibration values on a running Hangprinter without re-uploading firmware.

```
M665 W-1164.31 E-143.53 R998.78 T585.33 Y-114.98 U-977.11 I518.88 O-105.60 P2874.87
```

If you have `EEPROM_SETTINGS` enabled you can save these values with `M500`. If you don't save them they will be forgotten when you power down your machine.

Debug

The script accepts a `-d` or `--debug` flag. It calculates the difference between the output anchor positions and your manually measured ones:

```
Err_A_Y:   -52.307
Err_A_Z:   -28.532
Err_B_X:    28.781
Err_B_Y:    35.332
Err_B_Z:     0.022
Err_C_X:    -7.113
Err_C_Y:   -31.124
Err_C_Z:     9.399
Err_D_Z:     9.869
Method: L-BFGS-B
RUN TIME : 1.43998289108
```

For the `Err_ABCD_XYZ`-values to be meaningful you must have inserted your manually measured values into the `anchors` array in the script:

```
    # Rough approximations from manual measuring.
     # Does not affect optimization result. Only used for manual sanity check.
     anchors = np.array([[  0.0,   ay?,   az?],
                         [  bx?,   by?,   bz?],
                         [  cx?,   cy?,   cz?],
                         [  0.0,   0.0,   dz?]])
```

The debug check is only relevant if you suspect that the script outputs bogus values. Error larger than ca 100 mm is generally a sign that something's up.

Alternative Optimization Algorithms

The script accepts a `-m` or `--method` argument. Try for example

```
python ./simulation.py --method SLSQP -d
```

... for the `SLSQP` method that is faster, but requires more data points than the default `L-BFGS-B` method.

If you want to use the `PowellDirectionalSolver`, you also need Mystic:

```
git clone https://github.com/uqfoundation/mystic.git
cd mystic
python setup.py build
sudo python setup.py install
```

For more on usage, try

```
python ./simulation.py --help

usage: simulation.py [-h] [-d] [-c] [-m METHOD]
                     [-x XYZ_OF_SAMP [XYZ_OF_SAMP ...]]
                     [-s SAMPLE_DATA [SAMPLE_DATA ...]]

Figure out where Hangprinter anchors are by looking at line difference
samples.

optional arguments:
  -h, --help            show this help message and exit
  -d, --debug           Print debug information
  -c, --cx_is_positive  Use this flag if your C anchor should have a positive
                        X-coordinate
  -m METHOD, --method METHOD
                        Available methods are L-BFGS-B (default),
                        PowellDirectionalSolver (requires a library called
                        Mystic), and SLSQP. As a shorthand, you can use 0, 1,
                        or 2, for referring to the three methods respectively.
  -x XYZ_OF_SAMP [XYZ_OF_SAMP ...], --xyz_of_samp XYZ_OF_SAMP [XYZ_OF_SAMP ...]
                        Specify the XYZ-positions where samples were made as
                        numbers separated by spaces.
  -s SAMPLE_DATA [SAMPLE_DATA ...], --sample_data SAMPLE_DATA [SAMPLE_DATA ...]
                        Specify the sample data you have found as numbers
                        separated by spaces
```

Version #2
Erstellt: 2026-06-05 15:42:26 CEST von Mario Voigt
Zuletzt aktualisiert: 2026-06-05 15:50:08 CEST von Mario Voigt