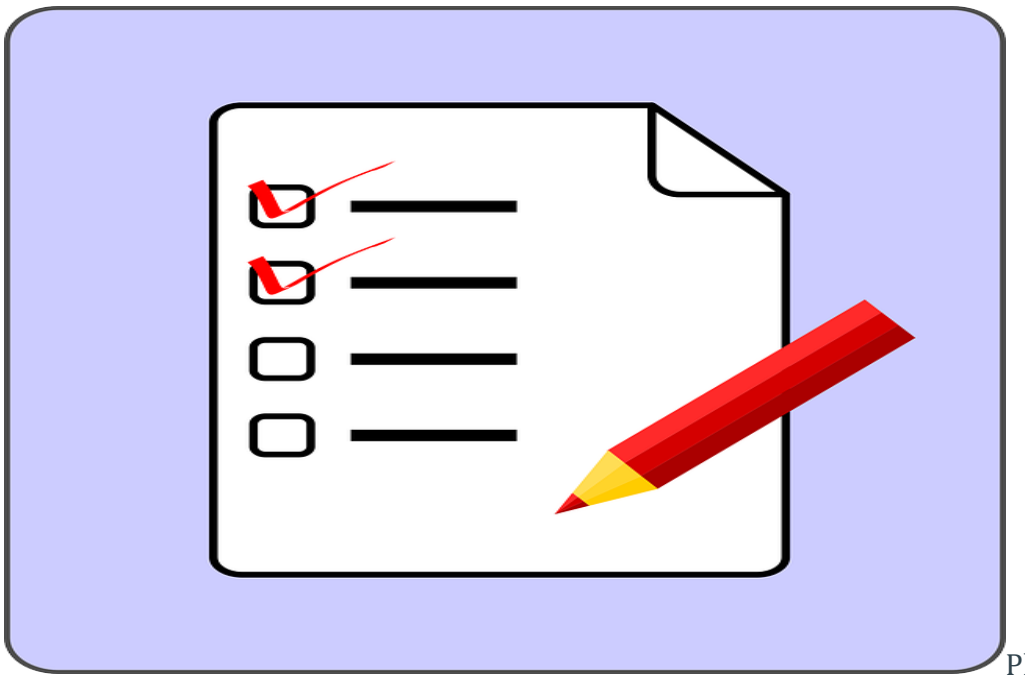
Hangprinter v4 Manual

Status Of This Document

This documentation might still have gaping holes here and there. If you're building a v4, say hi and connect with other builders in the discord channel.

Table of Contents

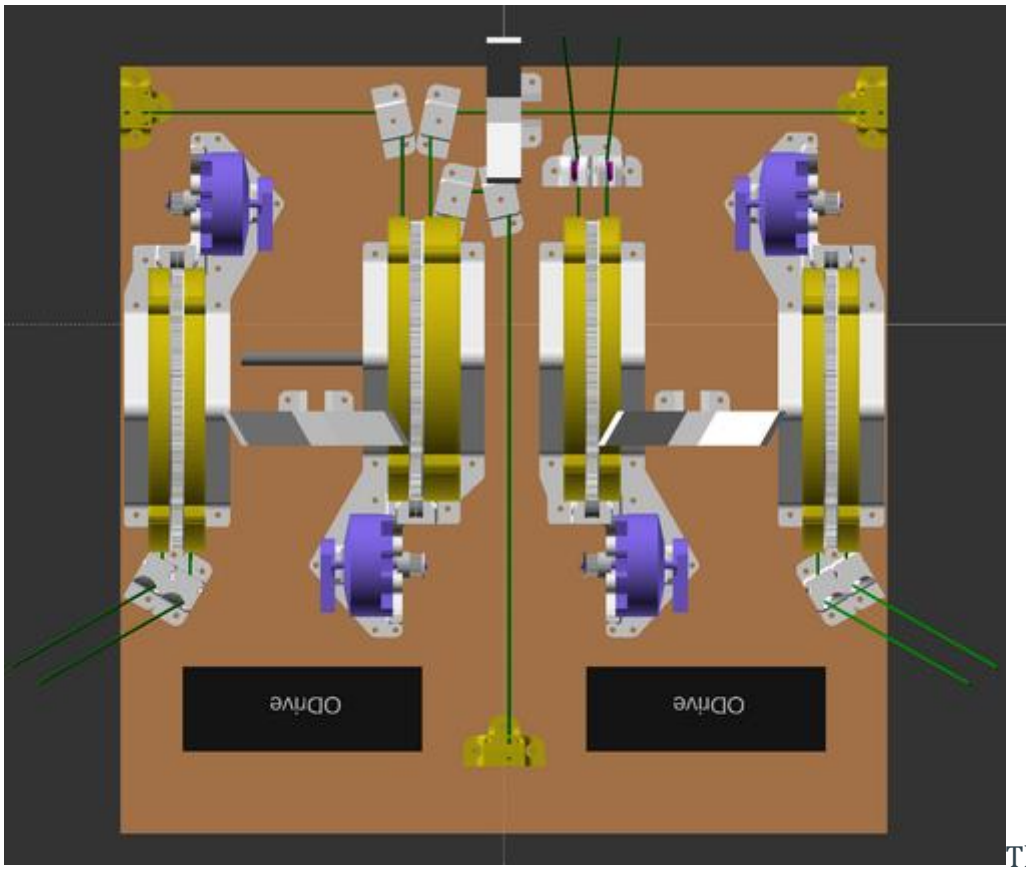
Sourcing and Preparing Wiring Loom



Please see the bill of materials (a Google Docs spreadsheet).

Hardware Assembly

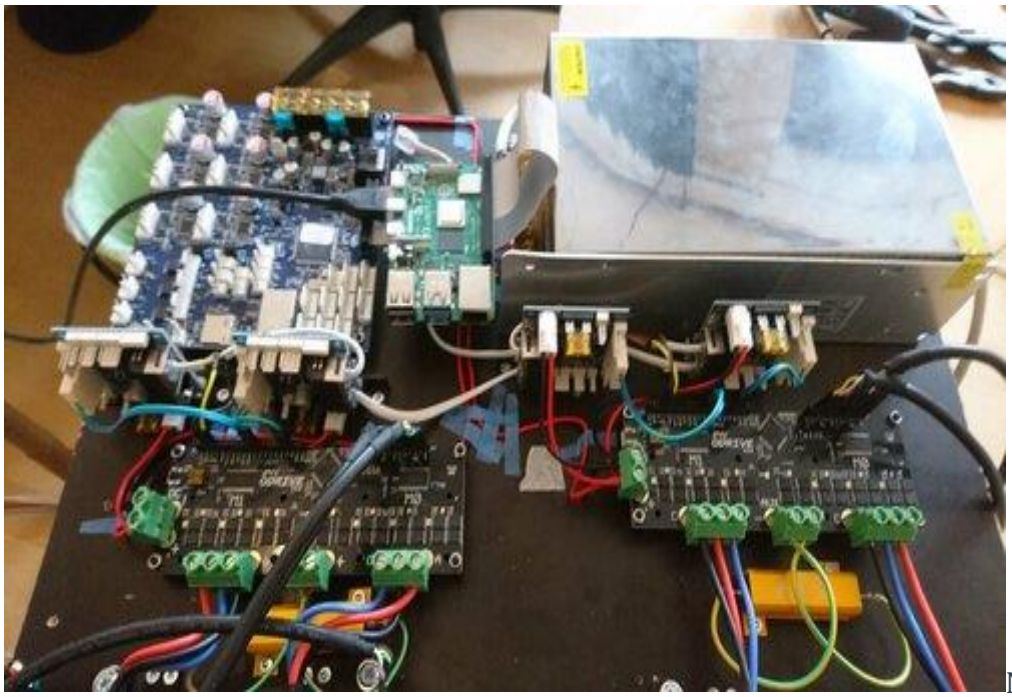
In the repo, there's an Openscad file called `layout.scad`. Open that file with Openscad. It shows all the parts, including where and how they're supposed to be mounted.

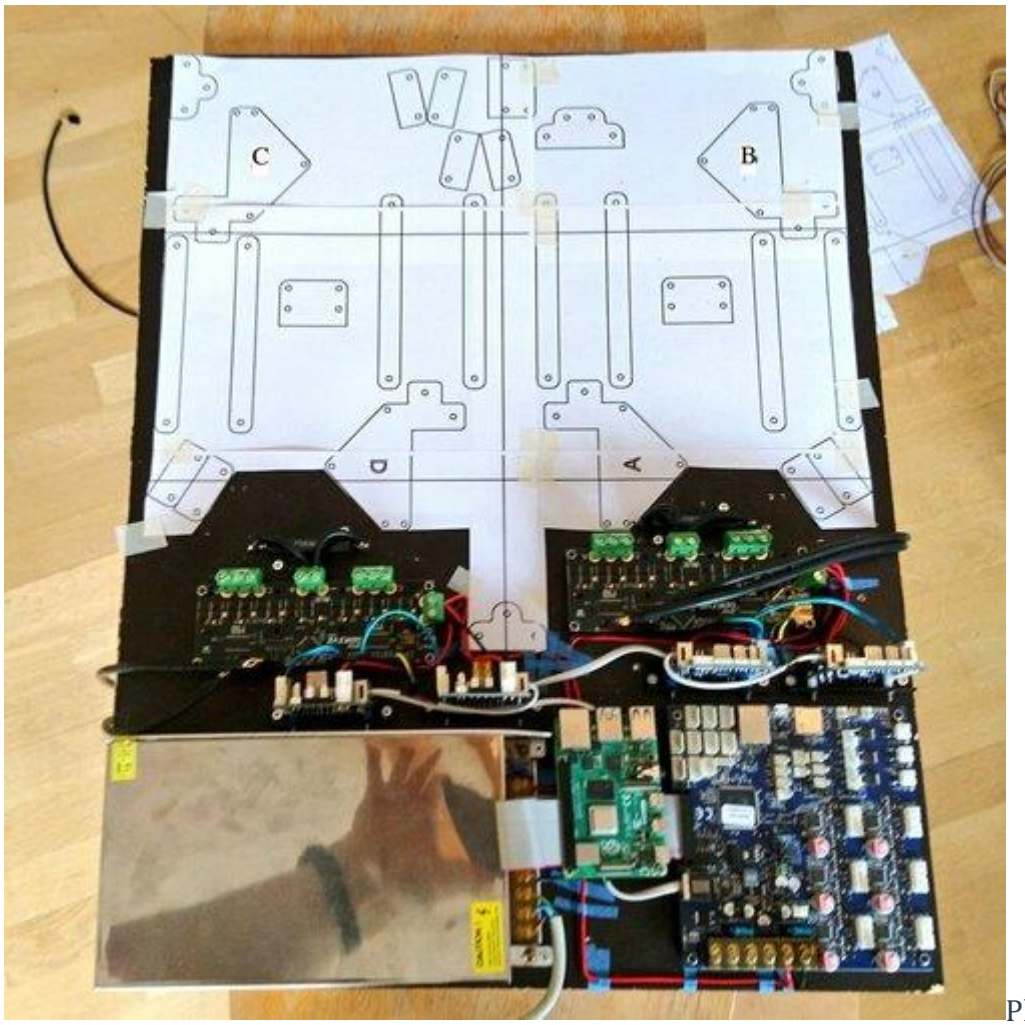


This view of the ceiling unit greets you when you open the layout.scad file with Openscad. Rotate the view (inside Openscad) by clicking and dragging with your mouse. Zoom by scrolling your scroll wheel.

To help positioning parts on the ceiling unit, 2d print out the layout_a4.pdf and use it as a template.

As for the circuit boards and the power supply, place them out in a configuration similar to this:

 Note the order of motor wires: Black, red, blue from left to right. Because of a limitation in RepRapFirmware for the time being (Sep 2021), we must connect the motor wires in exactly that order. Please click the image to open a larger version. Works for all images in this manual.



Please click the image to open a larger version. Works for all images in this manual.

Hot tips for circuit board placement:

- Use the spacers shipped with the ODrives also for the Duet3.
- The Raspberry Pi uses spacers that are twice as long, which may be 3d printed (see pi_mount stl in the repo).
- Make sure that the Duet3's USB port and SD slot are not covered by a 1XD card.
- Make sure the Pi's HDMI1 port is not covered by the Duet3.

Wiring

Warning: The motor wires must be connected in the exact order shown above. Otherwise torque mode will run backwards and create chaos. Also, auto-calibration won't work because signs will be reversed.

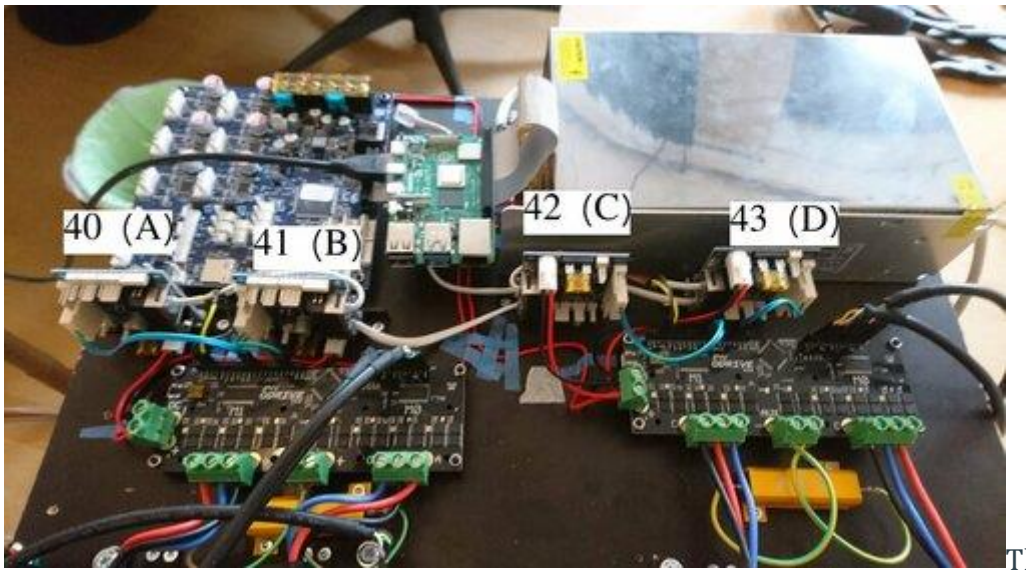
For all other wiring see

- Duet 3 docs,
- 1XD expansion board docs,
- ODrive docs.

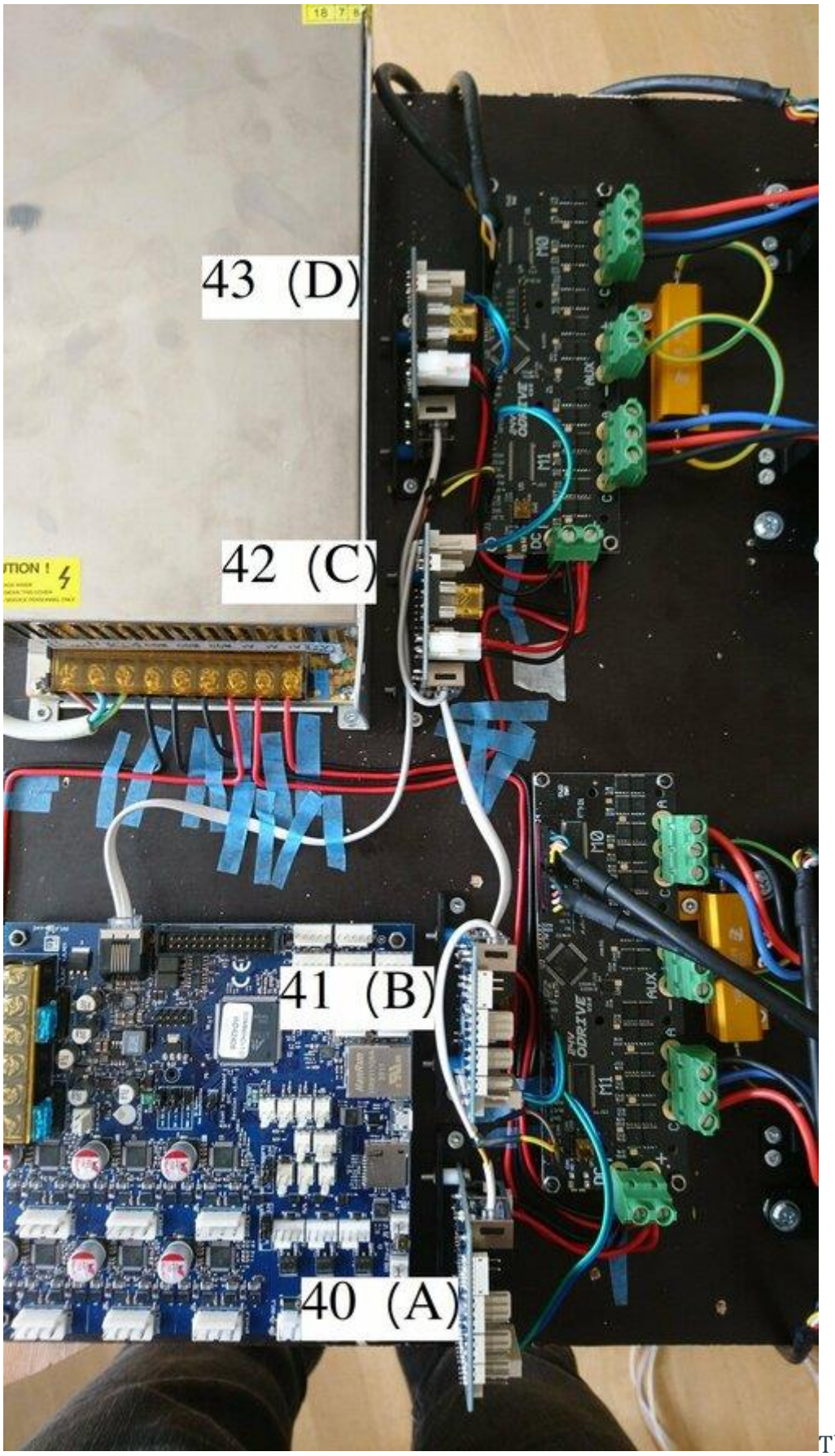
The Two CAN Buses

The Duet3 has two CAN buses coming out of its RJ11 (6P4C) contact. These two CAN buses live separately, in two different pairs of wires.

The two centermost conductors in the 6P4C contact carry the main CAN bus. It transfers step/dir signals from the Duet3 to the 1XDs.



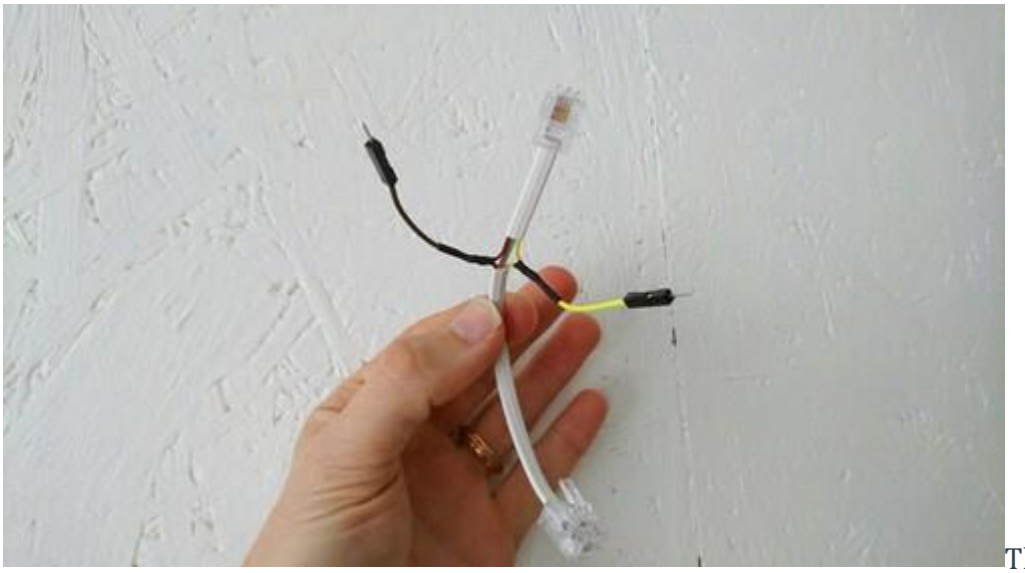
The main CAN bus goes inside the white wire, from the Duet3 to board 43, to board 42, to board 41, to board 40. Board 40 should have termination jumpers on. The other 1XD boards should not.



The main CAN bus goes inside the white wire, from the Duet3 to board 43, to board 42, to board

41, to board 40. Board 40 should have termination jumpers on. The other 1XD boards should not.

The rightmost and leftmost pins are not in use in 6P4C contacts. So we have two remaining conductors that are not used by the main CAN bus. The two remaining conductors carry the ODrives' CAN bus.

 The jumper wires carry the ODrives' CAN signal from the telephone cable into the ODrive.

The ODrives' CAN bus also goes inside the white wire, at least on my machine. It goes from the Duet3, through board 43, branches off into one ODrive, the other branch continues through board 42, through board 41, and ends in the second ODrive (left one on the image below). The 1XD boards does not and can not read the ODrives' CAN bus. It just passes straight through the 1XD boards.

With the ODrive CAN bus wired up, make sure you set the termination resistors correctly:

There are little switches inside the red circles, that flip CAN termination resistors on and off.

Mounting

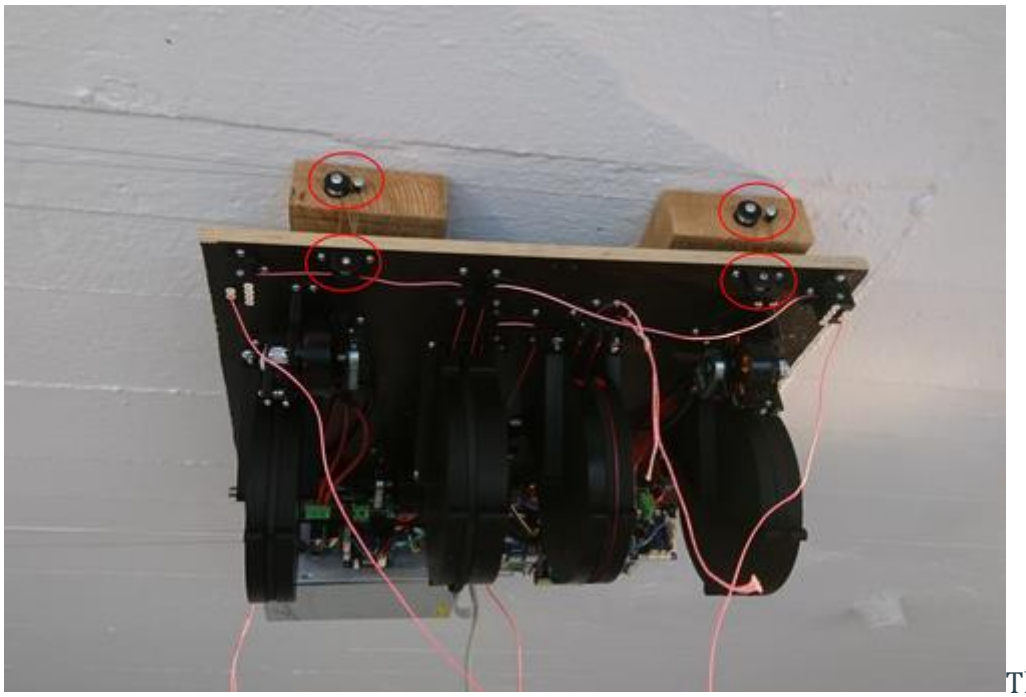
This part hasn't changed between HP3 and HP4, so I'm linking to old HP3 documentation for now:

- HP3 Build With tobben & Thomas Sanladerer (link directly into mounting)

Make sure anchors are rigid. Also, make sure your lines form nice Parallelograms (two pairs of parallel sides).

Optional Hoisting System

Screwing the ceiling unit onto the ceiling can be a bit challenging. I therefore added a hoisting system to my own HP4. See it in this tweet and this video. The little CAD models I used are shared here.

 The optional hoisting system. On my own machine I've routed the line back down to the ceiling unit one more time compared to what I had on this image.

Firmwares and Configuration

Stock RepRapFirmware will work in the future, but for now, I publish the RepRapFirmware I use here: link. Stock ODrivefirmware works for us out of the box.

Please look closely at my config files:

- config.g for RepRapFirmware
- configure_odrive.py for ODrivefirmware

Calibrating Anchors and Spool Buildup

This has changed a lot between HP3 and HP4. A computer vision system called hp-mark has been developed to assist, and largely automate, the calibration process. The system has been built and proved, see this video:

Replicating What's Going On In That Video

Ok, this will be a bit messy. hp-mark is still very much in beta, and requires the user to be comfortable doing a few things via the Unix terminal. Our main goal is to be able to run the script called get_auto_calibration_data_automatically.sh.

The computer vision system is called hp-mark. It consists of

- A camera with LED lights around it,
- A Raspberry Pi who controls the camera and the LEDs,
- Six retro-reflective markers,
- A computer program called hpm
- A main computer who gets images from the Raspberry Pi, and uses hpm to analyze the images.

See the hp-mark repo, and in particular the `README.md` and the `doc` directory for more guidance on how to set up hp-mark. The rest of this section will describe how to use hp-mark correctly with Hangprinter v4.

The get_auto_calibration_data_automatically.sh script is executed on the main computer. It expects your camera-connected Raspberry to be available at the ip called `rpi` in your `/etc/hosts` file. It's also assuming your Duet3 connected Raspberry to be called `duet3` in your `/etc/hosts`.

It's fine if `duet3` and `rpi` are actually the same Raspberry Pi board. Just configure them to the same ip address.

The Raspberry Pi needs the camera to be connected and calibrated (as described in the hp-mark repo), and it needs ssh to be enabled. It also needs a version of raspistill that matches your lens, or else your images' colors will be tinted. To get the one I use (matching my Arducam lo-distortion 45 deg lens), do

```
$ ssh pi@rpi
$ mkdir -p repos
$ cd repos
$ git clone https://github.com/ArduCAM/NativePiCamera.git
$ cd NativePiCamera/bin
$ chmod u+x raspistill_CS_lens
```

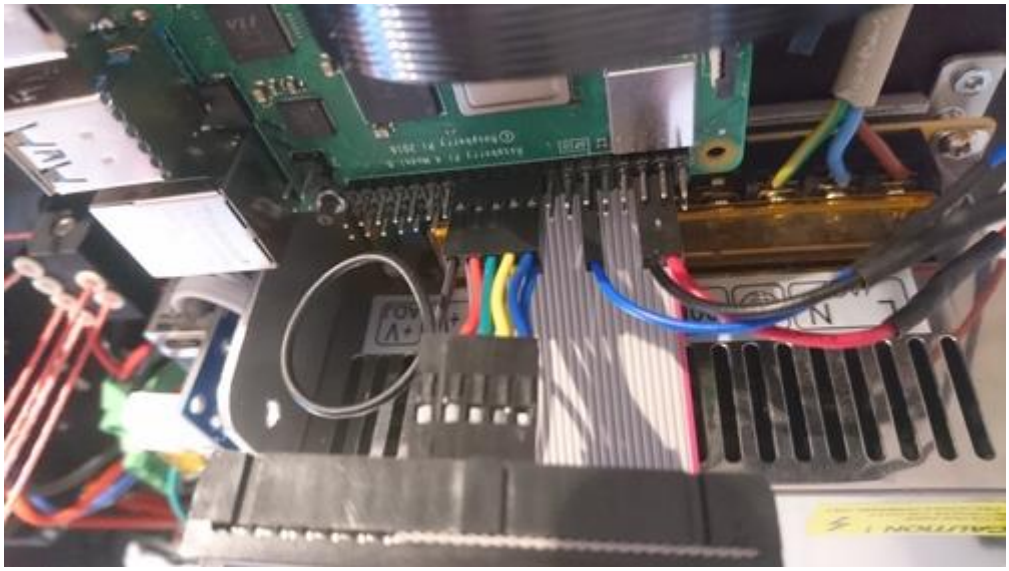
The duet3 official image is one comes with the camera disabled by default. To enable it do

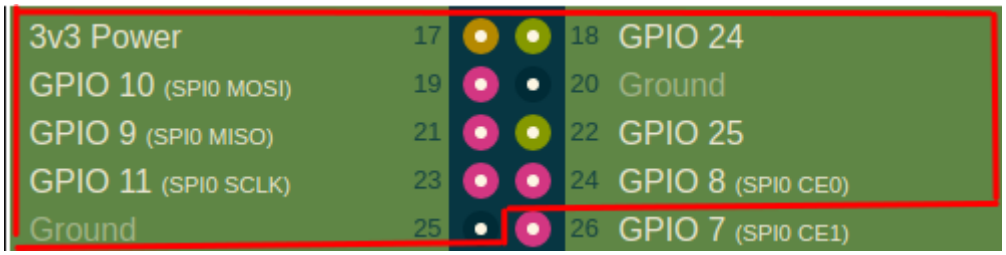
```
$ sudo raspi-config
$ [Interface Options] -> [Camera] -> [Enable] -> [Reboot yes]
```

We also have some LED rings around the lens (I have 20. Don's use more than 20, since the Rpi's 5V pin can't output much current). To light up the LEDs, I used these pins on the camera-connected Raspberry:

If `rpi` and `duet3` Are One And the Same Board In Your Setup

The Duet3 doesn't use any of pins 4, 6, or 12, although its connector covers them up. To work around the connector, you can created a distance between Rpi and the connector with nine short jumper wires like this:





Connect the Raspberry's pins 17-25 with the Duet, as they would have been connected through the standard connector. Power the Raspberry Pi with a separate wallplug power supply. The official Raspberry Pi Power supply is recommended.

Anyways...

Here's how you get code that lights up the LEDs:

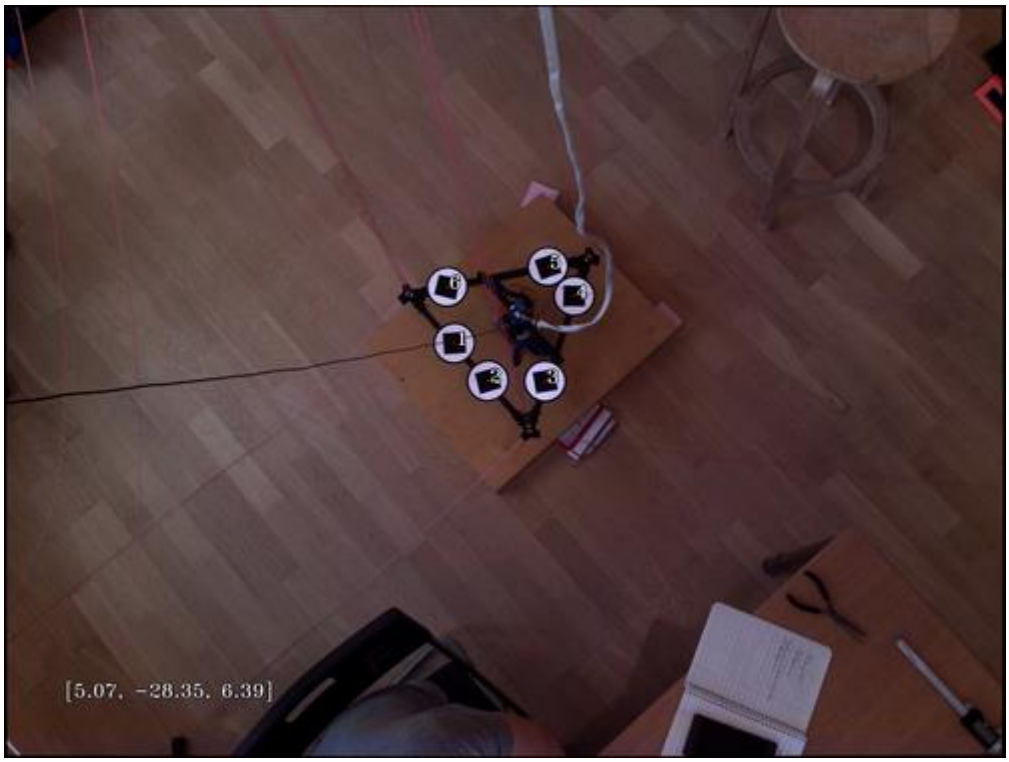
```
$ ssh pi@rpi
$ mkdir -p repos
$ cd repos
$ git clone https://gitlab.com/tobben/rpi_ws281x.git
$ sudo apt install python3-pip
$ sudo pip3 install rpi_ws281x
```

Test If Camera Usage Works

On your main computer, try to take an image and analyze it:

```
$ cd <path-to>/hp-mark/use
$ ./use_ssh.sh --show result
```

If everything worked, the LEDs should have flashed, an image should be taken, downloaded, analyzed by your hpm program, and the result (an image) should be shown on the screen, like this:



The console output should be similar to this:

```
$ ./use_ssh.sh --show result
/home/pi/repos/hp-mark/use
Captured image remotely: /home/pi/repos/hp-mark/use/images/DiUBh.jpg.
Copies home:
DiUBh.jpg 100% 4512KB 4.9MB/s 00:00
/home/tobben/repos/hp-mark/use
Will execute:
../hpm/hpm/hpm ../hpm/hpm/example-cam-params/loDistCamParams2.xml
../hpm/hpm/example-marker-params/my-marker-params.xml
./images/DiUBh.jpg --show result
[5.22761, -28.5536, 6.5542];
```

Double Check Torque Mode

The get_auto_calibration_data_automatically.sh script will not only use your camera. It will also put your motors in torque mode. It assumes the Torque_mode macro has been installed into your macro folder on the Duet3.

Play with the torque mode script in the web interface before running the full calibration script. For example, in the web interface, go to the console and to set motor A in torque mode, with 0.02 Nm of torque. Feel the spool gently with your and and confirm that the motor tries to pull line inwards onto the spool

```
M98 P"/macros/Torque_mode" A0.02
```

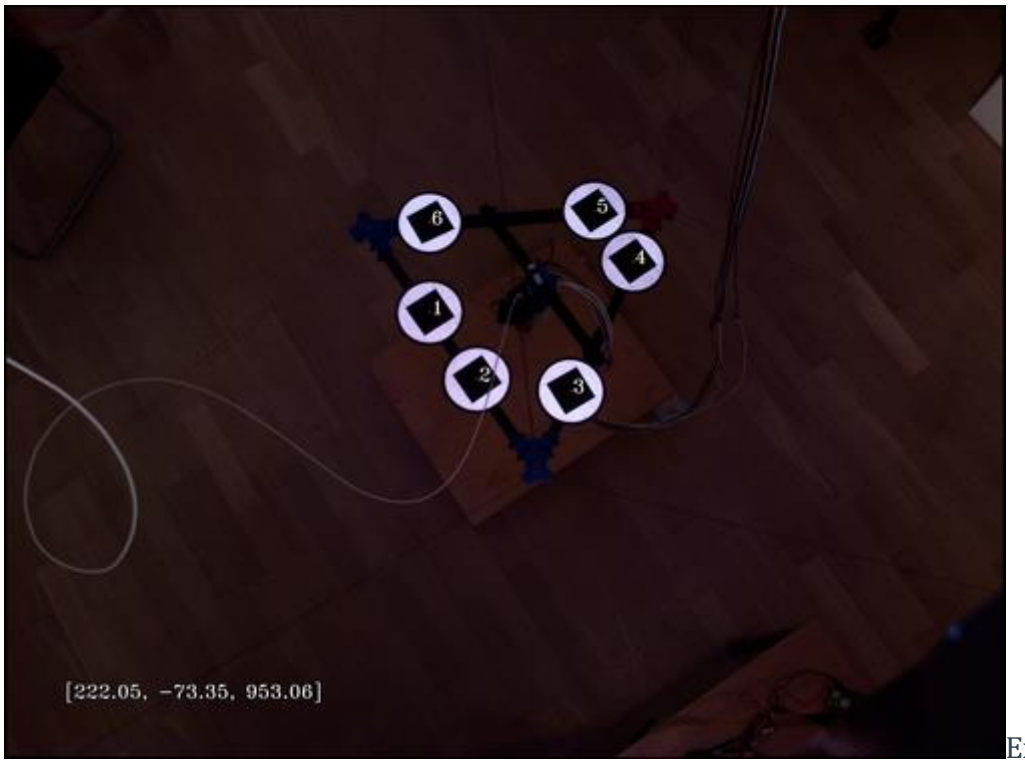
Repeat for all motors ABCD.

Getting To The Actual Calibration

With camera and torque mode macro in place, if all the stars align, it should be possible for you to collect the calibration data like this:

```
$ cd path/to/hp-mark/use
$ ./get_auto_calibration_data_automatically.sh --show result
```

The `--show result` option will make hpm stop and show you each measurement when it's done, and wait for you to press Enter before it continues.



Example of a successful data point collection with the `--show result` option enabled.

After collecting 18 data points (don't worry if a few were unsuccessful or printed a warning), it prints out the data it has collected in a format that copy/paste friendly for the next script we're going to use: simulation.py.

The output of simulation.py needs to be adjusted by yet another script, called just script.m.

If again, all stars align, you can re-run simulation.py as described inside script.m, and you'll end up with a perfect set of M669 and M666 commands, to copy/paste into your config.g.

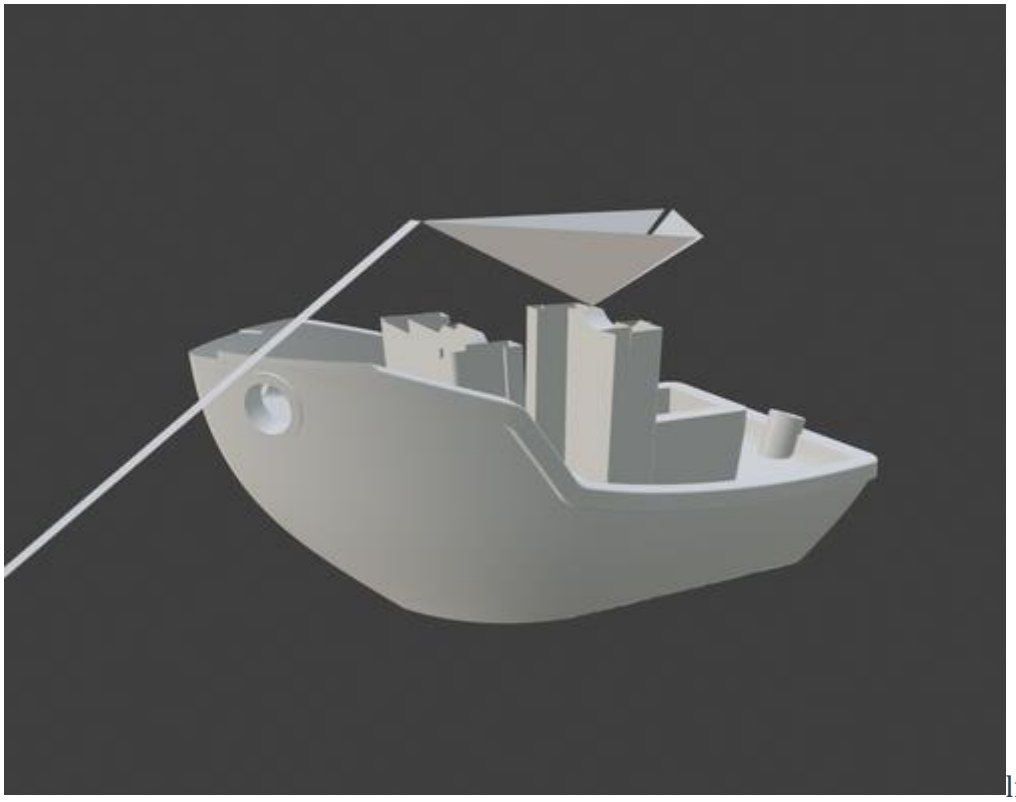
For anyone who have reached this far: I salute you. Reach me via Discord and tell me you need the auto calibration stuff, and I'll up-prioritize automating further and documenting better.

Slicing and Usage

I've published the Prusa Slicer configs that I use at gitlab.com/tobben/prusaslicer-configs.

Avoid Line Collisions

Before you start a large print, it's recommended to check if your model fits the print volume or not. This is done with line-collision-detector, a tool that is developed specifically for Hangprinter build volume verification.



line-collision-detector spits out a debug stl like this one upon detecting collision (if you asked for it).

Please note that many slicers will auto center the model before slicing it. line-collision-detector will not do that. Since you probably want to check if the centered version of

your model collides with lines or not, it's recommended to first import your model in the slicer, and then export it from the slicer as an stl, and then run it through the line-collision-detector.

Final Words

Building, mounting, calibrating, and running a HP4 is a big undertaking, and many of the steps are sparsely documented, but you are not alone. Be sure to check out the resources, there are some quite good ones.

If you spot an error or a missing link in the documentation, then please fix it and contribute the fix back to the repo. But also, do come by the Discord and say hi, or have a chat via Gitlab merge request or issue.

- tobben 👷