BED LEVELING EXPERIMENTS WITH KLIPPER

Gary Dyrkacz   February, 2025

Introduction

This discussion is a supplement to the larger discussion created as a record of transforming a stock Sovol SV06 with the original Marlin software to open source Klipper software, using a BigTreeTech Pad 7 as the secondary mcu. The original conversion file can be found in this repository.

This discussion expands on the bed leveling process, because of confounding issues that developed while trying to produce uniform first layer prints. No matter what typical procedures were applied: bed shimming, z tramming, X Axis Twist compensation, and bed mesh calibrations, all failed individually or in combination to produce uniform first layer prints. This was despite the ability to get bed mesh heatmaps that had a standard deviation of ~0.04 mm and a minimum to maximum range of 0.12-.14 mm.

The problem discussed is based on a single SV06, and could be due to a unique set of mechanical and physical issues with the printer. On the basis of other users in forums, I suspect I am not alone. Some of the problem could be my own lack of understanding of the mechanical, electronic, and software inherent in all printers. However, I believe I devoted enough effort to show that there can be black box type to the bed mesh leveling process that are not always resolved by the standard bed calibrations.

This discussion has also gone through several iterations, ending up with attempting to use two different DIY" "touch" probe methods complementary to the standard inductive probe. The final result is still confusion as to the fundamental cause of inadequate first layer prints based on standard bed calibration methods.

A hybrid bed leveling process was devised that consistently produces uniform first layer prints. This hybrid process involves direct intervention in manipulating bed mesh values in the *printer.cfg*. The process requires substantial initial user time investment. The advantages only become apparent with the frequency needed to reaffirm the bed mesh for bed flatness.

The final leveling process is spreadsheet dependent to quickly deal with manipulation of bed mesh values. Although direct manipulation of the *printer.cfg* bed mesh values is possible, the calculations would be tedious.

Much of the work displayed and discussed here is based on Excel to manipulate data, using formula's and potentially vba macros for more complicated data calculations, and for directly providing a method to relatively quickly change out the *printer.cfg* bed mesh block. The calculations are not sophisticated; any spreadsheet application will work.

In the directory with this document, is a heavily parred down version of the original Excel file that was used to scrape, record, and manipulate data. Missing are several Excel files from a torturous path dealing with the myriad exploratory phases of data reduction and calculation ideas. Even though the final worksheet process is streamlined, it is not a simple plug and play process. Effort will be needed to understand what some of the worksheets are doing, though in some cases there has been effort to describe the general operations on a worksheet.

For completeness, the entire bed leveling process of going from an OEM bed as shipped to the final process of getting good first layer prints is reviewed, before getting to the real solution, because these processes are all entwined.

The details presented apply strictly to an Sovol SV06 with the standard inductive probe. Much of what I discuss is valid for other 3D printers, but there are so many variations that especially with respect to the macros, you must check the code to make sure your printer is appropriate.

There is one more reason for this document. Often, I found myself reading and relying on other Klipper users' discussions. Some of these discussions were very sophisticated, clearly by expert Klipper users and software writers. Often the issues discussed were important, but often information was in the form of suggestions and not explicit examples. For an expert-to-expert discussion that is fine, but it leaves us unsophisticated users stumbling in at least partial darkness. To some extent, the discussion here will suffer some of this same problem. It was not designed for the new user who bought a printer last week. Especially how I resolved some macro issues using tuples and lists may help the intermediate macro writer avoid the huge time I spent fighting to make macros work.

Z TRAMMING

The original document describes the initial SV06 set up to align the bed with the extruder gantry. The process will not be repeated here. Because of an incompatibility of the BTT Pad 7 tramming code with the SV06, the classic soup can method was used for gross alignment, followed by the available Z_TRAMMING macro available online to do final tweaking of the bed leveling. This macro probes two Y axis centrally located X axis points symmetrically near the edges of the bed, and provides tweaks to the z screw stepper couplers in "hours" and "minutes" to correct the relative alignment of the bed with the gantry. This macro works well for the SV06.

What was personally objectionable was the use of just two points to determine the tilt of the bed. For instance, if one of the points, happens to be uniquely different height or highly biased relative to most of the points across the x axis, all the bed heights will be thrown off, and lead to uneven first layers. Of course, this can be remedied by changing the X and Y axis points in the macro code, to a region that is less prone to bias.

I decided to attempt a modified approach. A macro was created that allows a grid of points to be taken and stored as tuples of lists, which are then used to obtain a more refined Z tramming process. Unfortunately, the development process was not straightforward. As a neophyte macro writer, I found myself in coding hell with issues that apparently are considered too advanced in tutorials on writing Klipper macros. I have written a few macros, one of which will be discussed later, but this was a different level of coding. I will describe why coding was not straightforward, in the hopes that other non-expert Klipper macro writers may suffer less in handling such elements as tuples in lists and dealing with PROBE command measurements. Maybe an expert code writer might improve on my efforts, which may not be the most efficient to the point of being lame. The only thing that was certain is that it works consistently for me.

There were two main issues: One was the need to store and read back a list of coordinate data and Z axis PROBE command data tuples. Connected with this was learning how to work around the issue of how Klipper reads a macro, creates the operation code, but does not iterate over the original macro.

Initially, in the hope of saving time, I tried to get Google Gemini and ChatGPT to scope out the code. Both were a little help, but both AI's hallucinated, leading me down false paths, which took a lot of time to resolve. At least part of the reason the AIs failed, was that both fell back on jinja2 functions for answers. Klipper does not fully support jinja2. There were specific bits of code that Gemini did help resolve, often by asking very specific questions. Python coding might have bypassed some of the grief with jinja, but that involved another layer of learning on how to integrate python code into Klipper macros.

I succeeded, but not as cleanly as I would have liked, but I did not feel like expending even more hours on a macro that may have limited interest, except for me. As already suggested, someone else will find elements of the code useful to work with lists and tuples.

A final point about the code: A number of discussions and examples show the use of using global variables with loops. When Klipper runs a called macro, the macro is fully evaluated. If a loop is contained within a macro, any variables or printer state accessed inside the loop are static. The variables can be used inside the loop, but without some other mechanism will not be available outside the loop. To get around the problem, the general approach is to create global variables to store the data so other macros can use it. The Z_TRAMMING_GRID macro uses this method. However, the problem extends to even using any information created in the loop within the same macro. The way around this, after much frustration and searching is to use namespaces to handle information within the same macro. Refer to the macros in the folder for how namespaces were used.

The new z taming macro is, Z_TRAMMING.GRID, in the *.cfg* file with the same name in the current directory. There are several parameters that should be reviewed and adjusted.

- x_points: 5; readings/row between left offset and grid_size
- y_points: 3; number of rows to read (best practice is to pick an odd number)
- spacing: 20; distance between each row starting with the row at the center of the bed
- x_gap: 52; distance in mm from the left screw to where the nozzle is considered at 0
- z_screws_distance: 325; distance between the left and right screws
- screw_pitch: 4; 4 tpi is typical, but some printers use 8 tpi

The last two variables are specifically applicable to an SVO6, and will be different for other printers.
The output is fairly complete. All the points are probed first. For each Y axis row, the most important information will be a slope and intercept, the difference in mm at the right screw, and the hours and minutes to turn the right screw CW or CCW for each row. Generally, if the bed has been properly leveled with respect to bed screws and shims, the changes in each row will be nearly the same. As with the original Z_TRAMMING macro, the maximum difference across the bed tolerance is set to 0.04 mm, which at the z axis screws, for the SV06 becomes a maximum tolerance of 0.07 mm.

In practice, both tramming macros were used. The original Z_tramming macro runs faster, and is especially good as the first macro to try, especially for a badly aligned bed. However, the Z_TRAMMING macro tends to approach the minimum tilt in several test cycles, while Z_TRAMMING_GRD provides a more directed single cycle reference. Of course, the grid spacing can be altered to provide a wider or narrower region. The Z_TRAMMING_GRID resembles more closely the venerable soup can method of aligning the gantry with the bed. In essence, the soup can method is a gird approach, albeit two circular averaging grids.

Temperature and Bed Leveling
The Klipper documentation recommends bed leveling with the bed heated. The SV06 Marlin bed leveling process sets the bed temperature at 60 °C and extruder temperature at 120 °C. In early efforts to deal with a warped bed, calibrations were at room temperature, resulting in days of frustration trying to get even near a level bed. While trying to understand all the reasons for bed mesh levels that did not result in good first layer prints, I did a number of tests using the Klipper PROBE command under different temperature scenarios: Room Temperature, E@RmTB60, E120B60, and E195B60 (E=Extruder; B=Bed). The results showed large bed mesh differences at between room temperature and bed-only heated to 60 °C, versus E120B60, and E195B60. The bed mesh done at E195B60 showed no statistically different probe values than the E120B60 bed mesh.
The heatmap table below is the result of subtracting the average of two bed mesh calibrations done at ~17.5 °C from the average of two bed mesh calibrations done at E120B60,

204.996	-0.154	-0.119	-0.092	-0.075	-0.072	-0.077	-0.097	-0.113	-0.144	-0.174
183.552	-0.105	-0.078	-0.062	-0.047	-0.045	-0.050	-0.071	-0.077	-0.114	-0.141
162.108	-0.069	-0.047	-0.034	-0.022	-0.021	-0.029	-0.036	-0.033	-0.038	-0.063
140.664	-0.035	-0.017	-0.006	-0.001	-0.003	-0.008	-0.009	-0.010	0.003	-0.006
119.22	-0.019	-0.004	0.007	0.012	0.011	0.004	0.010	0.011	0.015	0.024
97.776	-0.006	0.008	0.015	0.021	0.021	0.015	0.021	0.020	0.029	0.033
76.332	-0.011	0.006	0.014	0.016	0.016	0.010	0.020	0.026	0.021	0.024
54.888	-0.013	0.003	0.011	0.016	0.012	0.008	0.014	0.022	0.002	0.010
33.444	-0.028	-0.009	0.002	0.010	0.010	0.008	0.015	0.016	0.000	0.013
12	-0.029	-0.005	0.013	0.019	0.022	0.021	0.028	0.029	-0.011	0.011
Y↑; X→	**27**	**47.333**	**67.666**	**87.999**	**108.332**	**128.665**	**148.998**	**169.331**	**189.664**	**209.997**

(The "heatmap" is created using Excel's Conditional Formatting function.) The especially large negative differences across the back of the bed correspond to a downward warp of the bed with increasing temperature. Overall, clearly the variations, with the exception of the back side are not very large. Still, the temperature data confirmed bed mesh calibration with the Marlin/Sovol SV06 recommended temperatures of 120E/60B was appropriate. Of course, this result is very printer specific.

First Shimming and Bed Mesh Calibration Attempts

The naïve, final bed mesh before running a first layer print was the result of a shimming process involving a combination of aluminum shims and judiciously placed 10 mm and 20 mm Kapton tape strips interspersed with Z tramming and manual adjustment of the z axis screws to achieve a reasonably uniform and flat bed parallel to the extruder x travel direction. The process was:

1. Heat bed to E120B60; wait a few minutes to stabilize.
2. Use Z_TRAMMING and/or Z_TRMMING_GRID macros to make changes by either adding metal shims to bed screw positions or Kapton tape to bed surface, or simply rotating appropriate z stepper motor feed screw couplers.
3. Once satisfied with initial tramming, get the default bed mesh, using BED_MESH_OUTPUT PGP=0, or copy it from the *printer.cfg save_config* section and paste the mesh into Excel.
4. Examine mesh data, assess what additional shimming is needed. Adjust the Z feed screws as directed by calculation (see section of revisiting tramming below). . (If necessary, use G Code *M84* to disable steppers to be able to move the Z axis couplers.)
5. Heat the printer 120E/60B (usually from mainsail). Wait a minute or so for temperatures to stabilize.
6. *Home All (G28)*
7. *BED_MESH_CALIBRATE*
8. Examine Heatmap to see what further tweaks need to be made.
9. Repeat steps 2-8 until bed is acceptably flat.

What is considered an acceptably flat bed is somewhat of a personal preference of what amount of time can be tolerated tweaking bed areas. The true measure is when first layer prints produce smooth surfaces without blobbing or open lines; this will be in a small range of values at each point. Mathematically, an initial assessment of flatness can be judged by watching the standard deviation and range of values of the bed mesh. In my typical case, the range of values in my later efforts was ~0.14 mm, and a standard deviation of 0.027. (Although the heatmap shows the range, the standard deviation comes from the worksheet.

All data was based on bicubic interpolation with 10x10 mesh, probe count=3; pps 4,4; tension 0.5.

To some extent, the shimming process is free-wheeling. It rarely can be followed strictly. Rarely were shim changes and Z axis screw adjustments made simultaneously. Changing too many variables at once easily leads to just confusion, on what changes were really accomplishing, especially as the bed gets closer to a flat surface.

Below is heatmap generated in Excel, of what I thought would be the final mesh at the time. Clearly, in this “optimized” bed mesh there are still probe points particularly at the bed periphery, with fairly large negative regions. The central area was expected to show only very small differences from zero based on using the average slopes to make changes to the z tilt, and z offset with probe at center of bed. The color scale runs from blue, -0.15 to red, 0.15.

205.00	0.016	0.010	-0.010	0.013	-0.037	-0.040	-0.065	-0.078	-0.084	-0.027
183.56	0.001	-0.008	-0.039	-0.020	-0.048	-0.047	-0.073	-0.079	-0.086	-0.036
162.11	0.034	0.018	-0.023	-0.005	-0.042	-0.043	-0.076	-0.076	-0.068	-0.030
140.67	0.039	0.032	-0.007	-0.002	-0.043	-0.041	-0.068	-0.069	-0.067	-0.025
119.22	0.045	0.030	-0.008	-0.007	-0.061	-0.053	-0.073	-0.067	-0.059	-0.015
97.78	0.036	0.031	-0.005	-0.001	-0.047	-0.041	-0.072	-0.062	-0.067	-0.018
76.33	0.020	0.010	-0.028	-0.018	-0.055	-0.045	-0.070	-0.060	-0.068	-0.018
54.89	0.011	0.013	-0.046	-0.044	-0.076	-0.063	-0.085	-0.073	-0.075	-0.040
33.44	-0.039	-0.042	-0.100	-0.086	-0.104	-0.094	-0.104	-0.091	-0.081	-0.091
12.00	-0.035	-0.054	-0.112	-0.100	-0.130	-0.109	-0.117	-0.111	-0.104	-0.092
Y↑; X→	**27.00**	**47.33**	**67.67**	**88.00**	**108.33**	**128.67**	**149.00**	**169.33**	**189.67**	**210.00**

(Note: this bed mesh is not the best routinely achieved in later efforts, but it is representative of the pattern of variation. Z tramming can be very different process with different printers. Use the recommended processes for a specific printer.

I was quite pleased with this bed mesh, until I ran the first 3x3 first layer grid test print. Layer thickness was set to 0.200 mm. Despite all the effort to achieve a flat mesh with shimming, z tilt adjustments, and X Axis Twist Compensation, the first layer results were disappointing. The mesh produced some leveling compared to turning off the bed mesh correction, but consistently, on multiple tests, the mesh calibration produced first layer prints that unequivocally showed severe differences. Despite meticulous effort with the Z tramming macro to level the bed before generating a bed mesh, so that the variation from the left side to the right side was less than 0.01 mm, the left side squares of the 3x3 pint always showed more squishing, and in some cases evidence of the start of blobbing. Meanwhile, the right-side layer squares always exhibited gaps between some or all the lines. Not only could the differences be seen, but even gently rubbing a fingertip or fingernail over the squares, the differences could be felt. These results were consistent for many prints and minor tweaks, no matter how much shimming and tramming variations were done. Some micrographs displaying the issue are in the original document file.

Whatever bed mesh calibration is supposed to do; it was not working as anticipated or expected with the SV06 with the Pad 7. One relatively new process that has been added to Klipper is X axis twist compensation. I initially though this might have something to do with my lack of attaining uniform first layer prints.

X Axis Twist Compensation

I was somewhat skeptical of this process, although some users claimed great benefit for printers like the SV06. Despite what the Klipper docs say, the twist range is not based on the nozzle position, but the probe position. So X had to start at 27 and end at 193. Y at 20 and 200. I ran AXIS_TWIST_COMPENSATION_CALIBRATE and AXIS_TWIST_COMPENSATION_CALIBRATE AXIS=Y. Here are the test results:

```
X axis: AXIS_TWIST_COMPENSATION_CALIBRATE: Calibration complete, offsets: [0.03555555555554646, 0.03430555555558179, -0.0698611111111278], mean z_offset: 1.567454
 Y axis: AXIS_TWIST_COMPENSATION_CALIBRATE AXIS: Calibration complete, offsets: [0.02027777777777917, -0.00513888888883919, -0.01513888888895252], mean z_offset: 1.585407
```

Except in the case of the X axis, the variations were quite small. The bad news was the prints with and without this compensation were effectively the same as before, with no visible effect on the final first layer prints. I ended up not using the X Axis Twist compensation.

Final Process to Produce More Even First Layer Prints

The variations in bed mesh are quite small. In early work, the full range of values fell within 0.19 mm for this particular effort, about the thickness of the first layer setting of 0.20 mm. The standard deviation of typical bed meshes was ~0.04 mm, which represents 12% change of the layer height for 67% of the values. However, that is within a single mesh.

At this point, some printer users, might be saying "overkill", in further effort to obtain a flatter bed. Instead of trying to fix the bed, just increase the first layer thickness, or just ignore it, especially since the worst variations were at the bed periphery. Basically, live with it. However, considering the supposed accuracy of the inductive probe, why should I not be able to achieve a more uniform bed everywhere?

The final solution involves direct changes to the section of the *printer.cfg* file, which is not recommended to make changes to. MY OWN WARNING: Anyone considering doing what is discussed below, better be comfortable with spreadsheet manipulations, handling *.csv* files in spreadsheets, and thoroughly familiar to what the SAVED_CONFIG section of *printer.cfg* is all about. By the end of several months work to change from Marlin to Klipper and resolve the first layer problem, I had developed a sense of how Klipper interacts with *printer.cfg*, at least to be confident enough to manipulate the bed mesh section. Of course, saving a backup copy of the starting *printer.cfg* file as a fall back, is a good idea.

There are several assumptions and criteria that need to be maintained to do this. The printer must be in a stable configuration. If different printing runs show widely varying bed meshes or first layer prints, there are likely flaky mechanical, electrical, or software issues with the printer; these must first be resolved:

- Check that every screw is tight that you can access.
- Check all electrical connections are tight.
- Make sure belts have the right tension.
- Make sure the bed is scrupulously clean. No glue residues. No fingerprints. (I prefer to wash with warm water with a bit of dish soap, followed by rinsing, drying, and finally wiping down with 100% isopropanol.

Do several calibrations and average the results as the base bed mesh to work from. Typically, the average error is approximately +/-0.02 mm. However, differences are typically higher at the edge, and smaller in the mid region of the bed. I also found that before each calibration, it is critical to home all axes first. When I did not do this, I found major changes to the bed mesh values. Generally, the bed mesh surface pattern was the same, but the absolute position values changed by as much as 0.13 mm.

The process to achieve flat, smooth, properly filled first layering prints, is to manually change the bed mesh values in *printer.cfg*. The process is not complicated, but is tedious. As previously suggested, the only reasonable way of doing this involves manipulating data using a spreadsheet. Just about any spreadsheet app will work, but I have a long history of working with Excel and vba macros.

This was a repetitive, several hours long process of adding or subtracting small increments of 0.01-0.02 mm to selective mesh probe points, especially on the left and right side of the bed, and running 3x3 first layer prints. The largest value changes to mesh X probe matrix rows 1 and 2 or 9 and 10. Many of the initial corrections were made across the entire Y axis, but some final tweaks were to individual probe points, especially at the corners. After a while, there is a certain intuitiveness to the magnitude of an alteration to make.

Although I chose to use a 3x3 matrix for speed on “roughing in” the corrections, a full first layer print or a 5x5 mesh was used in the final push to refine a compensator mesh. The time to print 25 squares pattern took ~22 minutes to print, a substantial increase in time, because of the increase from 9 to 25 squares.

The “trick” to this process was how the original bed mesh, based on multiple averaged bed meshes was adjusted. (If you are not familiar with spreadsheet manipulations, much of the following will be opaque, and require dedicated effort to learn workbook/spreadsheet work.) If familiar with spreadsheet operations, there are variations on these processing steps:

Get the coordinate positions of the nozzle and probe that BED_MESH_CALIBRATE is using:

1. In mainsail, run the Klipper macro, BED_MESH_OUTPUT PGP=1. Copy the entire output data to a worksheet.
2. Convert both sets of text data to cells:
    a. Highlight column with the probe text, Use Data| Text To Column with space, comma and other “|”, delimiters to convert the text to cells.
    b. Delete the first column of time stamps
    c. Filter the entire text positions column to remove residual time stamp text. Copy the filtered data below the existing data and clear the filter on the original data. If desired, the original rows can be entirely deleted at this point.
    d. On the newly filtered data, use Home|Find/replace to replace the remaining “(“ and “)” with nothing.
    e. Highlight all 4 columns; sort smallest to largest by col A, add a sort level and sort by Column B.
    f. Copy the filtered, sorted data to a new worksheet.
    g. Use the =UNIQUE formula to extract only the unique elements from the stream of data in each of the columns to a separate set of columns. Get the average of the differences between the position points to get the true probe jump distance between probed positions.

Extract the bed mesh height data:

3. With temperature set to E120B60, do a BED_MESH_CALIBRATE macros. At your discretion, save the data. Do BED_MESH_OUTPUT PGP=0. Ignore the initial data and copy only the bed mesh height values at the bottom of the output stream. Copy the data to a spread sheet. If instead of using BED_MESH_OUTPUT, the matrix in the *printer.cfg* file is copied, every probe value needs to be multiplied by -1, because the data is stored as the negative of what the heatmap would show.
4. Run at least one more mesh calibration, as in step 3 Do SAVE_CONFIG, if you believe this data may be better than previous, but it is not necessary. (Use heatmap to view output). Copy the data below the same previous set of data.
5. All the data copied to the worksheet is in text form. Convert it to columns with comma and space delimiters checked.
6. If using more than one bed mesh set, average the same probe points from each matrix to produce a final bed mesh matrix.
7. Using =AVERAGE(…), find the average of this bed mesh.
8. Lastly, create one more matrix which in which each probe points is subtracted from the average. For quick understanding of variations, Excel’s Conditional Formatting is applied. Typically, dark blue represents negative numbers, white is 0, and dark red is positive.

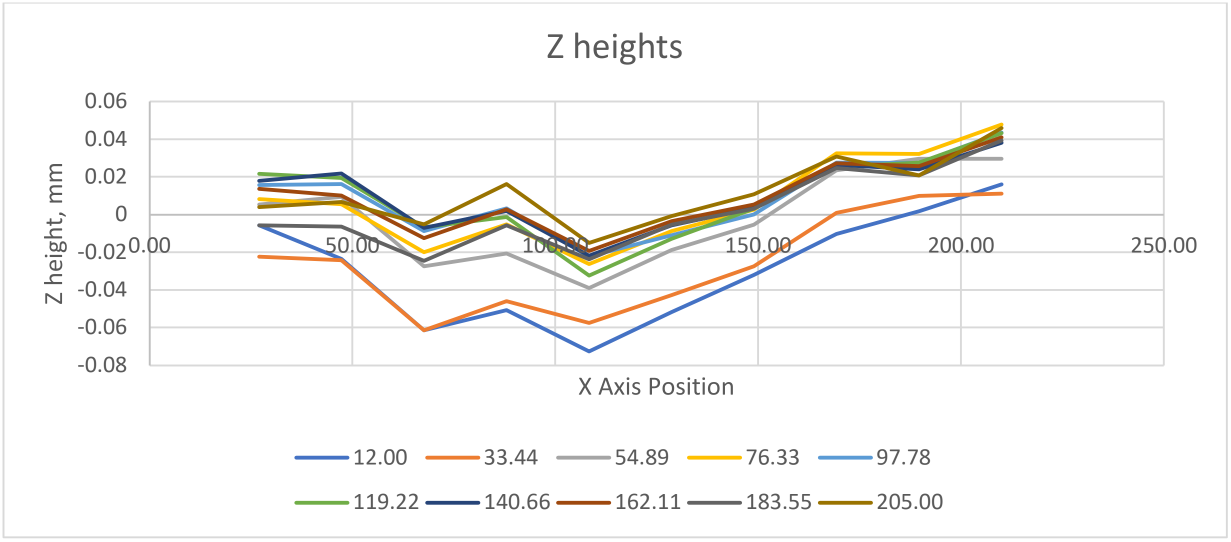
The resulting matrix is the base bed mesh matrix. Copy the values of this matrix to a separate worksheet. For reference, copy and paste the probe positions around this matrix as reference points. The top left position of the bed mesh is the front left side of the printer when facing printer. The top right position of the matrix is the front right side of the printer.

Below is an example of an average base bed mesh without any compensation

bed mesh averaged with any offset change at Z Offset:1.591 **color limits -/+0.15,**

Y↓; X→	**27.00**	**47.33**	**67.67**	**88.00**	**108.33**	**128.67**	**149.00**	**169.33**	**189.66**	**210.00**
12.00	-0.0058	-0.0236	-0.0615	-0.0507	-0.0726	-0.0518	-0.0321	-0.0103	0.0017	0.0160
33.44	-0.0224	-0.0242	-0.0614	-0.0460	-0.0576	-0.0428	-0.0275	0.0008	0.0099	0.0112
54.89	0.0053	0.0093	-0.0275	-0.0207	-0.0389	-0.0189	-0.0054	0.0235	0.0296	0.0296
76.33	0.0081	0.0054	-0.0201	-0.0053	-0.0263	-0.0088	0.0025	0.0324	0.0321	0.0479
97.78	0.0157	0.0162	-0.0088	0.0033	-0.0221	-0.0111	0.0000	0.0275	0.0274	0.0436
119.22	0.0215	0.0193	-0.0068	-0.0014	-0.0324	-0.0129	0.0031	0.0254	0.0276	0.0433
140.66	0.0178	0.0217	-0.0072	0.0019	-0.0219	-0.0058	0.0035	0.0264	0.0240	0.0382
162.11	0.0136	0.0100	-0.0126	0.0028	-0.0192	-0.0035	0.0053	0.0272	0.0256	0.0410
183.55	-0.0058	-0.0065	-0.0246	-0.0057	-0.0238	-0.0054	0.0031	0.0247	0.0208	0.0394
205.00	0.0040	0.0067	-0.0053	0.0161	-0.0150	-0.0008	0.0107	0.0308	0.0207	0.0460

Below is a plot of the z heights of individual rows of data,



Several points about this mesh. First, note that this mesh is different than the first, because it is averaged. No two meshes will be precisely the same anyway. Second, the probe axis positions have been reversed. This better represent how the data is represented in *printer.cfg* and is consistent with how data is downloaded from mainsail.

As already indicated, the colors and values represent z heights: positive values, signified by red hues, mean that the height of the bed is above the average height of the bed. This also can be referenced that at a specifically defined nozzle height, the distance between the bed and nozzle is smaller. Referencing to the average height of the bed, as near as I could find, is part of the Klipper bed mesh calibration process.

As already discussed, this bed looks good in principle, but did not produce uniform first layer prints. I will describe later my efforts to understand additional experiments to try to understand what was causing the issue. For now, I will describe the process to "fix" the bed mesh data to yield consistent first layer prints.

Bed Mesh Compensation
The approach is to build a compensation matrix (compensator) that can be added to the normal bed mesh matrix produced by BED_MESH_CALIBRATE macro to compensate for whatever causes the problem, even if the culprit is unknown.

Before developing the compensator, make sure the bed is well within the z tramming limit. It is also advisable to run the bed mesh calibration several times, copy to a worksheet by using BED_MESH_OUTPUT PGP=0, copying only the mesh data, and adding to the worksheet.

This compensation matrix is initially based on the inductive bed mesh, or better an averaged bed mesh, multiplied by -1, or simply starting from scratch with a zero compensation matrix. To this, values are added or subtracted until a satisfactory first layer print is produced at every square of the 3x3 print (and later the 5x5 print).

The process works by using a series of complementary visual techniques to observe defects on the individual 3x3 squares (or whatever first layer print is being used). For instance, by removing the print, and holding up to a light, gaps between lines or where the sides join the interior lines are immediately clear. This means that the extruder head was laying down plastic too high from the bed, despite what the original bed mesh correction indicated. If the distance is large enough the plastic solidifies before binding to the bed, and the nozzle ends up dragging the material across the bed. Obviously, in this case the printer is immediately stopped and a positive correction added. If not even one square can be printed, the problem may be the z offset is set to high. This is a constant issue with my SV06. Setting the z offset using the paper method, consistently produced a too high z offset by 0.20 to 0.04 mm, forcing me to manually increase the z offset in the *printer.cfg* "saved" section.

If blobbing is observed, a negative correction needs to be added. Lightly rubbing a finger over the surface was a surprisingly sensitive detection method for blobbing, even detecting very minor levels of surface blobbing when compared to evenly printed squares. Also, shining a small led light (in my case, a red single led that came with the printer) at a shallow angle, further emphasizes even the beginning of blobbing problems.

Another visually useful method was just observing the initial lines that were drawn by the printer around the periphery of each square and between squares. These single lines, immediately showed thickness variations that signaled where problems where height problems were likely to show up.

As the process continues, often more subtle corrections can be discerned. For instance, in some cases, only one side of a square would show open lines, or blobbing, suggesting adjacent probe areas needed to be adjusted.

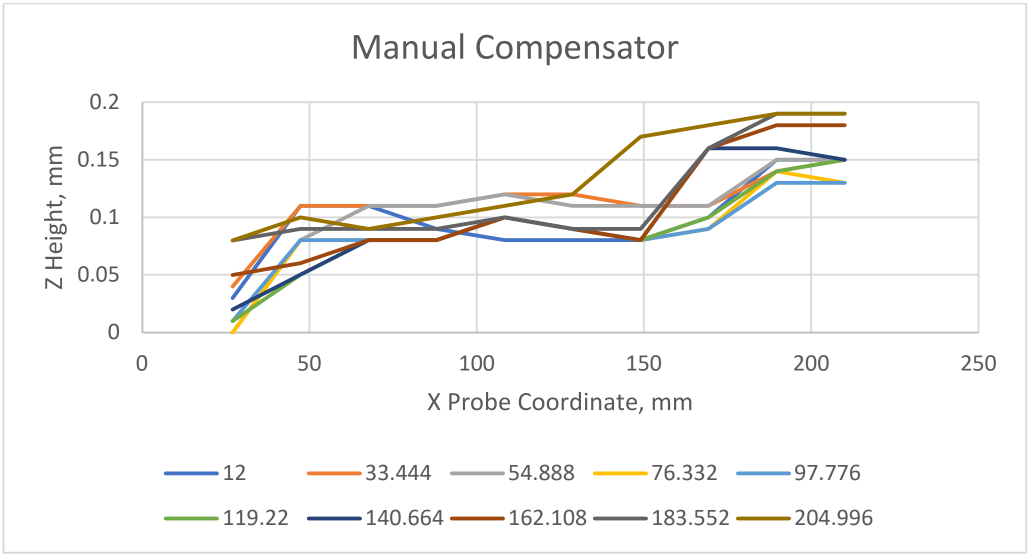
Finally, using a cheap digital microscope camera, to view the squares either face on or edge on helped show how uniform the print surface was; this was only used sparingly though.

Especially in the early stages, fairly large area corrections were made. Except in severe cases, of no plastic binding to the bed, changes of up to 0.04 mm were made. Large corrections are dangerous to make, and could result in the nozzle crashing into the bed. Typically, when the finest corrections need to made, the frustration has also built up with the time demand for this process, and the tendency to over-compensate can set in. Clearly, this is a time-consuming process, requiring at least several hours of work and repetitive printing.

As an example of a compensation matrix with my SV06, below is the correction matrix that is added to an inductive bed mesh to provide uniform first layer prints, with all nine squares.

Y↓ X→	27.00	47.33	67.67	88.00	108.33	128.67	149.00	169.33	189.66	210.00
12.00	0.03	0.11	0.11	0.09	0.08	0.08	0.08	0.1	0.15	0.15
33.44	0.04	0.11	0.11	0.11	0.12	0.12	0.11	0.11	0.14	0.15
54.89	0	0.08	0.11	0.11	0.12	0.11	0.11	0.11	0.15	0.15
76.33	0	0.08	0.08	0.08	0.1	0.09	0.08	0.09	0.14	0.13
97.78	0.01	0.08	0.08	0.08	0.1	0.09	0.08	0.09	0.13	0.13
119.22	0.01	0.05	0.08	0.08	0.1	0.09	0.08	0.1	0.14	0.15
140.66	0.02	0.05	0.08	0.08	0.1	0.09	0.08	0.16	0.16	0.15
162.11	0.05	0.06	0.08	0.08	0.1	0.09	0.08	0.16	0.18	0.18
183.55	0.08	0.09	0.09	0.09	0.1	0.09	0.09	0.16	0.19	0.19
205.00	0.08	0.1	0.09	0.1	0.11	0.12	0.17	0.18	0.19	0.19

The magnitude of some of the changes shows just how far off the original bed mesh was from producing good first layer prints.



The legend represents the corresponding Y axis probe value for each X axis probe points.

There appears to be stepwise changes over several of the X axis positions. To some extent, this was an artifact of the use of a 3x3 grid of test first layer squares. However, the pattern is more strongly influenced by the inductive bed mesh pattern itself.

Below is the final bed mesh calibration, which is the sum of the original inductive bed mesh and the compensator matrix.

Y↓ X→	27.00	47.33	67.67	88.00	108.33	128.67	149.00	169.33	189.66	210.00
12.00	0.0242	0.0864	0.0485	0.0393	0.0074	0.0282	0.0479	0.0897	0.1517	0.1660
33.44	0.0176	0.0858	0.0486	0.0640	0.0624	0.0772	0.0825	0.1108	0.1499	0.1612
54.89	0.0053	0.0893	0.0825	0.0893	0.0811	0.0911	0.1046	0.1335	0.1796	0.1796
76.33	0.0081	0.0854	0.0599	0.0747	0.0737	0.0812	0.0825	0.1224	0.1721	0.1779
97.78	0.0257	0.0962	0.0712	0.0833	0.0779	0.0789	0.0800	0.1175	0.1574	0.1736
119.22	0.0315	0.0693	0.0732	0.0786	0.0676	0.0771	0.0831	0.1254	0.1676	0.1933
140.66	0.0378	0.0717	0.0728	0.0819	0.0781	0.0842	0.0835	0.1864	0.1840	0.1882
162.11	0.0636	0.0700	0.0674	0.0828	0.0808	0.0865	0.0853	0.1872	0.2056	0.2210
183.55	0.0742	0.0835	0.0654	0.0843	0.0762	0.0846	0.0931	0.1847	0.2108	0.2294
205.00	0.0840	0.1067	0.0847	0.1161	0.0950	0.1192	0.1807	0.2108	0.2107	0.2360

To keep the heatmaps consistent, the range here is the same as in the previous heatmaps, but clearly some values exceed the positive limit.

Only rarely would printing to the edges of the bed be necessary, and this compensated approach gave more confidence in reducing some print lift off issues. As an added safeguard, fade was set in the [bed_mesh] section. Another advantage of this compensation process was that if the bed was washed with soapy water, dried, and finally wiped clean with 100% *iso*-propanol, there was no need to use any glue stick on the bed, even for first layer prints.

Unfortunately, the correction matrix serves only as a bookkeeping aid rather than a mechanism to apply to any bed mesh to flatten the bed. Only if z tilt, which in the present case is based on the relative gantry position, and the bed plate(s) do not change, is it possible to use the correction matrix in a reverse sense to a change made only to the extruder, such as a probe or nozzle change.

Bed Mesh Calibration Conclusion

The typical process of bed shimming, Z_tramming, X axis twist compensation, and BED_MESH_CALIBRATE function did not provide adequate first layer prints. A process was defined based on direct modification of the automatic inductive bed mesh data using a compensation matrix that is added to a regular bed mesh. The compensation matrix is based on visual evaluation of first layer prints. The process is time intensive, requiring up to several hours of work, depending on how far from flat the bed is. The advantage is that it is a lot faster than manually probing the bed at every point using the paper test method to determine z heights.

What's going on?

Before discussing the potential reasons for the lack of correction by the bed mesh calibration procedure, this forewarning in the Klipper documentation is appropriate:

"The Bed Mesh module may be used to compensate for bed surface irregularities to achieve a better first layer across the entire bed. It should be noted that software based correction will not achieve perfect results, it can only approximate the shape of the bed. Bed Mesh also cannot compensate for mechanical and electrical issues. If an axis is skewed or a probe is not accurate then the bed_mesh module will not receive accurate results from the probing process"

From <https://github.com/Klipper3d/klipper/blob/master/docs/Bed_Mesh.md>

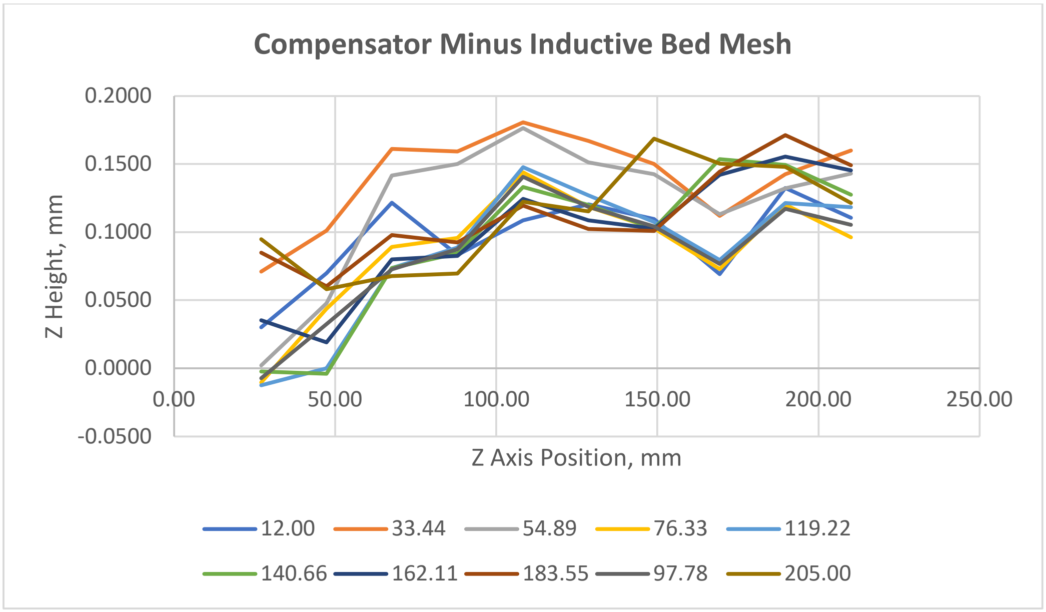
In the current case, the magnitude of the corrections needed at some positions, represents more than one half a layer height. I had expected that given a reasonable shimmed and reasonably non-tilted bed, the automated bed mesh calibration would adequately compensate for the differences. Clearly, it did not.

One perspective is that the SV06 is not an industrial strength printer, so I should just expect and tolerate more variations. However, print-to-print, calibration to calibration, the bed calibrations had a consistent precision and accuracy, much better than the needed manual adjustments that were ultimately made. My probe with 5 measurements at each point, leads to an overall standard deviation of +/- 0.0021 mm. The average error in my probe measurements was +/- 0.013 mm, which is ~10% of the worst variations observed in the correction matrix.

Temperature as the source of error was previously discussed. E195B60 AND E120B60 produced heatmaps that were very similar. Any deviations were within the expected error range, ruling out temperature as a cause for the discrepancies.

BED_MESH_CALIBRATE and X Axis Compensation did not adequately compensate for an uneven bed surface, even when adjusted for temperature. This is confusing, considering the precision of an inductive probe. Only manual, time-consuming, probe adjustment achieved an acceptably even first layer.

To get a better handle on the type of changes the bed was exhibiting, the original bed mesh was subtracted from the manually determined compensator. A plot of the X axis values as a function of Y values is below,



Subtracting the inductive bed mesh from the compensator, effectively distills out the compensation changes from a theoretically uniform, flat bed. With a couple of possible exceptions, the overall trend is nonlinear, which is not what would be expected for a simple X axis twist. The nonlinearity might be overlooked as just noise if the range of corrections was small, but that is not the case. Moreover, there are what appear to be three separate nonlinear regimes: the front of the bed, the back of the bed, and the middle regions

To get some kind of handle on what is going on, I found myself immersed in trying to understand just what is Z offset? Part of my initial confusion was the word "offset" being used in different contexts in different explanations. Each context has a great deal to do with printing, but little to do with each other. An excellent, brief description of the process is here:

How the bed leveling algorithm works?...Prusa Forum

From <https://forum.prusa3d.com/forum/general-discussion-announcements-and-releases/how-the-bed-leveling-algorithm-works/>

With a sensorless inductive probe, the bed mesh calibration goal is to find the point where the inductive probe reaches a height above the bed that satisfies the trigger voltage of the inductive sensor circuitry at various bed positions. This height is not directly the height needed to make a good first layer. This is one kind of offset, the bed mesh offset. This probe height is too high to do any printing.

The calibration process is over a matrix of bed positions, with the probe obtaining the trigger distance for every position in the matrix. The number of positions probed depends on what we consider the best and most efficient number to model the bed architecture for flatness. The bed mesh calibration software then generates a matrix of evenly spaced positions in the X and Y bed directions. Of course, the mesh positions are also subject to compensation for extruder endstops and xy "offsets" between nozzle and probe…another use of "offset".

Couple of other points about the calibration. The bed mesh points we see are not raw measurements. Once the points have been taken, the matrix is averaged, and each point subtracted from the average. (It took quite a while of Google searching to find this out.) I call this a mesh "normalization" operation, although using the term this way is not quite what is usually considered a normalization.

The bed mesh calibration process is even more complicated. The bed mesh needs to represent all the space on the bed. This means some sort of interpolation calculation is needed to figure out the bed mesh offset value between the often widely probed positions. I am not clear on just how many probed positions are included in the interpolation range. I suspect the equation is fitted over multiple probed points, where the range of points used is controlled by the *bicubic_tension* parameter (if using bicubic algorithm). I tried to understand the Klipper python code, but kept getting lost in the object coding stream and python module *#include* references. If the latter is true, then the original measured probed values will be modified according to how well the fitting equation fits the bed curvatures. In addition, any points outside the matrix range are assumed to be the same as the nearest last point read. (I suspect a forward-looking prediction algorithm, such as Kalmin filtering might be a better choice, but is more complicated to implement.)

The upshot of the interpolation procedure is that the final difference value representing the bed flatness is unlikely to remain as the originally measured value. Thus, it is possible some bed configurations may not adequately compensate for unevenness and can lead to under- or over-compensation relative to the "real" bed mesh variations.

Also involved in this interpolation procedure, is the incorporation of the mesh_pps, mesh points per segment values. We end up with a bed map of height differences relative to the Home bed mesh probe value for each position. The values are related to what is found in the SAVED_CONFIG section of *printer.cfg*.

However, that does not tell the printer where the nozzle needs to be, to create a decent first layer. The data only signifies that the bed itself has all sorts of problems and here is the value to compensate for whatever created the uneven bed. The second height "offset", the z offset, is where the magic happens to tell the printer how much lower to go from the bed mesh value to create a properly smooshed line to give an even first layer.

The reason for a G28 – "Home All", before it analyzes the bed surface is for the printer to establish the endstops where it was and where needs to be on all 3 axes.

There is a simple inverted relationship between the Z_offset and the bed mesh. It is possible to manually adjust the relative relationship between the two. For instance, if every point of the bed mesh is manually reduced by, say 0.03 mm, adding 0.03 mm to the *printer.cfg Z_offset*, keeps the relative differences the same. A print will show no differences. Although this relationship is generally not useful, I found that in some cases when altering many bed mesh points manually, it was useful to renormalize and use this relationship to optimize the bed to a bed mesh more centered around zero change. Also, the averaging calculation means that the center of the bed will likely not be zero. If the bed is reasonably flat, or at least with a minimal height asymmetry, then the bed center will be close to zero.

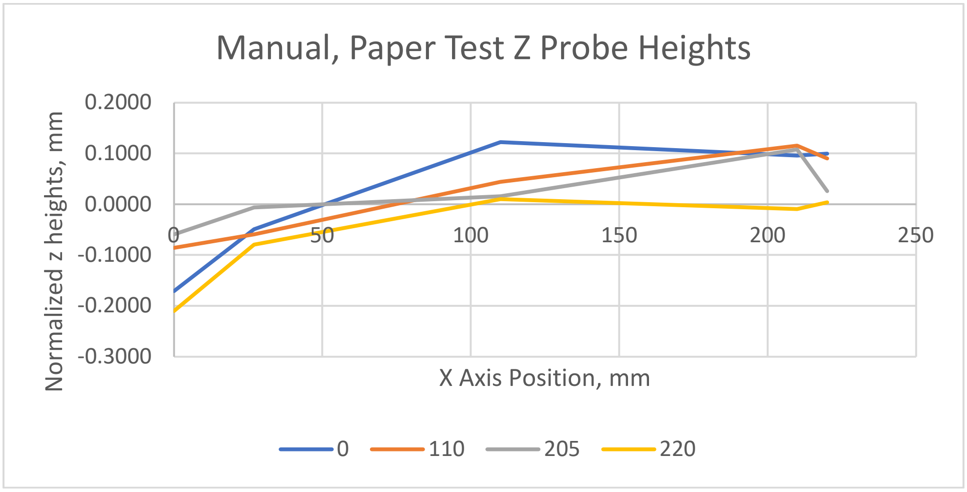
For practical printing, the distance at which the nozzle smooshes the extruding filament to also fill at least half way between the current line being printed and the next line to be printed. This inherently lowers the extruded line depth, but leads to even first layers. Generally, the necessary height above the real bed surface is going to be related to the first layer height set in the slicer. The bed and extruder temperatures also play a role. In the present case, the first layer height was 0.2 mm. I imagine smooshing should be around 0.1 mm, which is near the height of a piece of paper. In my case, the paper was 0.092 mm. The distance between where the bed mesh is triggered to stop, and where we need the nozzle to be to get good first layer prints is the z offset. This value is determined manually. It will always be a negative number, although it shows up in *printer.cfg* as a positive number.

Generally, this Z offset is determined at the same position as the probe Homing position. For instance, in Klipper, using the PROBE_CALIBRATE tool, the macro first reads the probe at the current user defined position, then based on the x and y nozzle offsets, moves the nozzle to the same position the probe just occupied, and waits for the user to manually lower the nozzle to find the optimum position using the "paper test" or feeler gauge. In either case, it is likely that a bit of tweaking of the z offset value may be necessary.

The X axis twist macro partially automates getting z heights at strategic points by moving to the pre-defined probe points, and directing the user to do a z height calibration test, using the micro adjusting menu. Once z offsets have been compiled the macro calculates a correction and if invoked in *printer.cfg*, automatically compensates for the twist.

Bypassing the automation, below is a grid approach of z height data, manually determined using the PROBE_CALIBRATE command:

Y ↓X →	**0**	**27**	**110**	**210**	**220**
0	-0.1709	-0.0494	0.1226	0.0956	0.0996
110	-0.0854	-0.0594	0.0441	0.1156	0.0896
205	-0.0594	-0.0064	0.0156	0.1076	0.0256
220	-0.2099	-0.0794	0.0101	-0.0094	0.0036



The data was determined by subtracting each raw height value from the average of the entire matrix of points, and then multiplying by (-1).

With respect to the X axis twist routine, note how critical the choice of calibration positions is. In the case of this specific SV06 bed, if only using the standard three points, different points separated by only 10 to 20 mm would have profound influence on the compensation. In some cases, the correction would be better fit by a nonlinear equation.

However, there are two issues with the data. First, the magnitudes of the compensation are different; this can be expected, based on how the paper tests were conducted. The heights were based on where the lightest touch of the nozzle to the paper resulted in noticeable friction. If desired, the entire matrix of points can be arbitrarily raised or lowered to better match the values of the compensator matrix. The exact value to match is not easy to discern, because there are clearly overall pattern differences.

The second problem with the data is the difference between the paper tests heights and the compensator overall patterns. Some of this data reflects a similar left to right increase as defined by the compensator produced by the first layer calibration method, but the two compensation patterns vary substantially at the back of the bed. The real reason was never investigated further. However, repeated paper test measurements show variations less than 0.02 mm for any point. What is still clear is that, as already discovered, the magnitude of the variations will not be fixed by an x axis twist process alone, because of the nonlinearity and variation of the bed along the Y axis.

At this point, temperature, bed mesh algorithm, and probably only x axis compensation seem to be ruled out as causes for the inability of Klipper to compensate for the true height variation the right side of the bed.

What is left? A lot of speculation.
Perusing various post on the internet, bed meshes not producing good first layer prints are most often due to mechanical issues with the printer. All the obvious mechanical and physical possible places, including belt tensions, were checked. Nothing obviously wrong was discovered. Of course, I could have missed something.

One possibility is that this is an emergent problem of a host of small differences adding up to produce the large anomalous bed mesh offsets. Certainly, the measurements exhibit errors, due to natural measurement errors, and inherent errors in the mechanical and electrical properties of the printer. However, if this was just a host of small errors, more inconsistency in the anomalous areas would be expected from run to run, which is not observed. This is also not consistent with the observation that most probed points do not require any manual intervention.

Is the use of the Kapton tape to shim the bed, altering the distance between the probe and metallic surfaces sufficient to throw off the probe heights? Not likely. Some regions with Kapton tape do not show any anomalous heights. The overall character of bed unevenness was encountered before any shimming with Kapton tape. If it does affect the probe measurement, the effect is very small. Without the tape, there is little chance of getting anywhere near a flat surface on the printer, with only five mounting screws for shimming.

Another speculation is that for the probe to function properly, the metal bed material itself must be uniform in thickness and in homogeneous in metal composition. Both these were unknown factors. The PEI coated top plate was checked for thickness variations on the left and right sides with a micrometer; the thickness was uniform probably within less than 0.001", so the bed plate itself cannot be the problem.

Another possibility: Some sort of complex interaction with the plates magnetic field. An inductive probe works by inducing a field in the metal bed, which in turn produces an eddy current that changes the energy field of the probe. The change in energy is detected and amplified by the sensor circuit. An inductive sensor is not affected by a static magnetic field. Of course, the SV06 uses a magnetic bed to hold down the printing plate. Does the inductive probe really see a static magnetic field on a magnetic plate? The sensor moves up and down to sense the eddy current that the metallic bed generates, and the probe circuitry senses with respect to some threshold. As the probe mechanically moves down to trigger the detection circuitry, it moves through the plate's static magnetic field gradient as it approaches the bed. Moving in a gradient field produces a change in magnetic field, in addition to the induced field produced in the metal. If the change is short enough and significant enough to interact with the probe's electronic oscillator circuitry and detection threshold, any variations in the uniformity of the bed plates magnetic field, might cause calibration issues.

Further work

This compensation issue kept bothering me. If the material or magnetic properties interacting with the inductive probe interactions are the problem, then using a different type of probe, such as a direct contact touch probe, might provide a more realistic bed mesh. The Sovol SV06 is my only printer, and I was not about to remove the inductive probe, go through the grief of purchasing and installing a different probe, with no guarantee that I would get better results. Instead, I decided to make my own touch probe. I have a competing interest in microcontrollers such as an Arduino and ESP32 modules and sensors, which helps. I am not an expert. I dabble.

The main criterion for the probe was to attach the touch probe to the extruder without taking the probe apart in any way. I ran across a couple of DIY touch probes, but they were fairly complicated to build and requiring partial disassembly of the extruder body.

Two probes were designed, but the two models are nearly equivalent. They differ in where the touch probe was located relative to the inductive probe and nozzle.

As for the probe mechanism itself, I had in my "good junk" pile an old computer CDROM drive. When I originally disassembled it, I noted the read/write head moved on a highly polished rod through a small bearing mandrel. After isolating it from the drive mechanism, I found the rod moved almost frictionless through the mandrel. The metal mandrel body was oddly shaped and far too large for the project, so was cut and filed down, and holes drilled to attach to a 3D printed mount.

Next problem was how to use the mandrel and rod to sense the bed height. I decided on two sensor modules, an ADXL345 module, and an optocoupler module as possibilities. Below is the image of the optocoupler used,

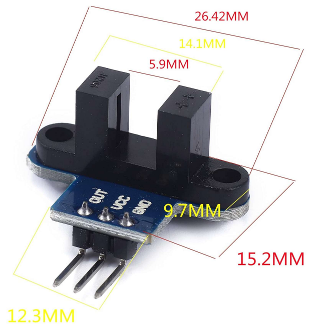


Figure 1. Optocoupler

This type of optocoupler is widely available.

The space on the extruder to mount the touch system is limited to the right side, with a "usable" width of ~18 mm. The top and bottom surface is also not completely flat. There are two hexagonal raised portions to

which the front extruder assembly is screwed. These are close to 1.7 mm higher. The mount had to be able to slide over these. In addition, the right side of the holder had a side apron to both help locking in the mount to the extruder body, and provide more rigidity to the entire holder.
The probe mount fitted with the optocoupler is shown in the photos:

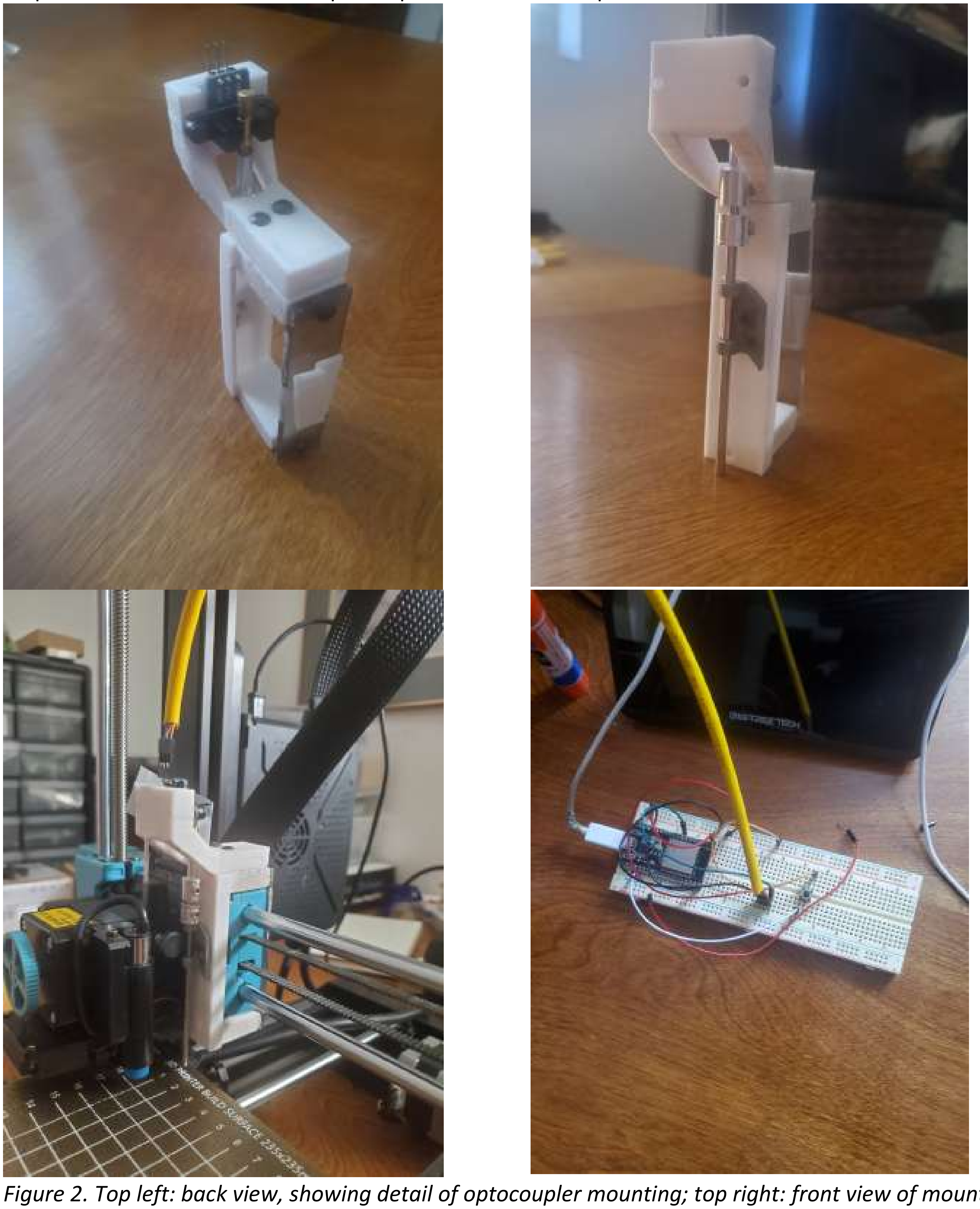


Figure 2. Top left: back view, showing detail of optocoupler mounting; top right: front view of mount with mandril and rod; bottom left: touch probe mounted on extruder; right bottom: ESP32 breadboard mounting with touch button to start ESP32 data logging.

The version of the touch probe mount in the photos was designed as two pieces to save having to do a reprint after the adxl345 probe tests. A Fusion *360 touch_probe2.f3d* model file of an updated, one-piece version is available in the current folder. The back plate was cut from an old piece of scrap plastic in the workshop. The mount "wings" are slightly shorter than the extruder depth to allow solid clamping. The two white rectangular plates taped to the back of the backplate are just printed shims that go on the top and bottom undersides of the holder, because the left end of the extruder has two raised areas to allow for screws holding the extruder together.

The top left photo clearly shows the mandrel and rod assembly that was cut out of the CDROM body. On the touch rod, are two aluminum adjustable collars. The lower collar acts as the height depth stop collar to fix the starting rod position above the bed. The second adjustable collar is more of an inverted piston shape with an extended shaft to act as the integral light blocker to trigger the optocoupler. (In this specific case shown, a smaller brass shaft was glued into the aluminum collar shaft, because the original shaft length was too short for this two-piece design.) This collar is adjusted to be just below the optocoupler trigger position. Not shown is the platform for the adxl flange that was a tight fit to the top collar extended shaft. The aluminum collars were cut on a lathe, and drilled and tapped for set screws, to lock the collars in place.

Optocoupler and ESP32 Measurements.
Both the adxl345 and the optocoupler modules can be used with an Arduino or ESP32 mcu. The optocoupler proved to be much easier to work with, especially for data analysis. Though it would have been nice to directly use the Raspberry PI Pad 7 adxl interface to acquire the data, or another available port, this involved an excursion into a whole new level of programming and interfacing, requiring far too long a learning curve.

I will not go into details about what an ESP32 mcu is, or the sketch code. There are many tutorials available. The sketch is straightforward to understand, if a reader is already familiar with an Arduino or ESP32 and writing sketches. The circuit setup is similarly very simple and will not be shown; again, there are tutorials for incorporating momentary buttons with an ESP32.

As can be observed in the photos, the optocoupler sensor module is screwed to the probe holder itself, and therefore does not affect the rod motion. When the rod blocks the LED light between the two towers, the module circuitry responds with trigger voltage, which the ESP32 registers. Querying the ESP32 microsecond timer before and after triggering, and outputting the time difference provides the probes contact movements. Of course, this means data recording rates are easy to contend with compared to the adxl345..

The offsets on the probe rod were: x_offset, 26 mm; y_offset, +4 mm compared to x offset 25, y offset -20 for the inductive probe. Therefore, the x offset is very similar, but quite different for the y offset.

The only real drawback is that the ESP32 needs to be manually triggered to start reading data. So, in addition to setting the ESP32 to read the data using an ISR, a momentary touch button was added to the circuit and the button and bounce code added to the sketch. The sketch, *ESP32_optcoupler.ino*, is in the folder. The Klipper macro that starts the printer side of the operation, *esp_matrix_probe.cfg*, is also in the folder.

The sketch is primitive. It uses the Arduino IDE Serial Monitor for printing the time data. The ESP32 was connected to a laptop USB port to supply power and take data from the ESP32. No external power source was needed. Comparing with the ADXL probe described later, the analysis of the output data is straightforward.

Once the Klipper macro is started, the printer will first "Home All". The extruder will then move to the left front left corner of the bed. During that movement is when the ESP32 button must be pressed to start data acquisition. Once all the positions have been probed, the button is pressed again to stop the ESP32 from reading information. Because only a limited number of runs were made, no provision was added for a micro sdcard to store data. The data from the Serial Monitor was copied to Notepad ++, saved as a *.csv* file, and brought into Excel on a desktop PC. There is one catch in copying the data from the Serial Monitor. The Arduino IDE being used is version.1.9.18, and not the latest 2.xxx version. The new versions, as of two years in, do not allow copying a long stream of values. The alternative is to use a terminal program to access the COM port attached to the ESP32.

Data processing is fairly simply. The output format is a stream of three values: time start, time jump, and time difference. The time jump data is only used to recognize when the probe has been triggered. There will be a large time jump indicating the extruder has traversed in an x/y direction, followed by two more rows of shorter duration (if using 3 z height probing). This pattern signifies an individual bed position probing. The data stream, is then rearranged using three simple cell formulas. For instance, if the first time value is in cell C49, then in cells G51 to I51 the formulas: =OFFSET($C$49,(ROW(G1)*3)-3,0), =OFFSET($C$50,(ROW(H1)*3)-3,0), =OFFSET($C$51,(ROW(I1)*3)-3,0) are added, respectively. The x and y position data is then added for reference and the time values are divided by two and manually copied and pasted into bed matrix form. Using the already determined mm/ųs value, the times values are converted to mm. The accuracy of this method was excellent. The 2σ average standard deviation was 574 ųs, corresponding to 0.003 mm.

As suggested previously, the height values were then normalized to the average of all the matrix values. A resulting bed matrix is below:

ESP32 Bed Height Data **color limits -0.15/+0.15**

Y↓; X→	**27.00**	**47.33**	**67.67**	**88.00**	**108.33**	**128.67**	**149.00**	**169.33**	**189.66**	**210.00**
12.00	0.1082	0.0277	-0.0043	-0.0100	-0.0216	0.0041	0.0213	0.0190	0.0391	0.0839
33.44	0.0830	0.0291	-0.0051	-0.0065	0.0020	0.0178	0.0219	0.0157	0.0274	0.0603
54.89	0.0650	0.0297	-0.0059	-0.0042	0.0160	0.0238	0.0186	0.0062	0.0112	0.0357
76.33	0.0628	0.0274	-0.0054	-0.0050	0.0048	0.0089	0.0053	-0.0176	-0.0138	0.0158
97.78	0.0369	0.0195	0.0035	-0.0048	0.0009	-0.0037	-0.0048	-0.0252	-0.0172	0.0361
119.22	0.0267	0.0088	-0.0086	-0.0205	-0.0228	-0.0073	-0.0091	-0.0288	-0.0306	0.0101
140.66	0.0223	0.0117	-0.0189	-0.0216	-0.0212	-0.0131	-0.0188	-0.0367	-0.0375	0.0116
162.11	0.0173	-0.0073	-0.0287	-0.0261	-0.0183	-0.0087	-0.0204	-0.0400	-0.0433	0.0215
183.55	0.0054	-0.0182	-0.0374	-0.0289	-0.0220	-0.0092	-0.0209	-0.0402	-0.0497	0.0149
205.00	-0.0034	-0.0266	-0.0406	-0.0293	-0.0329	-0.0237	-0.0275	-0.0468	-0.0522	0.0138

Comparing this mesh to the inductive mesh suggest the two are very similar. The data does not resemble at all the bed mesh matrix that produces the best first layer results.

Although the adxl data was not as reliable, indications were that the data was likely to establish the same conclusion.

Subtracting the inductive data from the ESP32 data produces:

Normalized ESP32 probe runs minus normalized inductive probe **color limits -0.15/+0.15**

	27.00	**47.33**	**67.67**	**88.00**	**108.33**	**128.67**	**149.00**	**169.33**	**189.66**	**210.00**
12.00	0.0058	0.0236	0.0615	0.0507	0.0726	0.0518	0.0321	0.0103	-0.0017	-0.0160
33.44	0.0224	0.0242	0.0614	0.0460	0.0576	0.0428	0.0275	-0.0008	-0.0099	-0.0112
54.89	-0.0053	-0.0093	0.0275	0.0207	0.0389	0.0189	0.0054	-0.0235	-0.0296	-0.0296
76.33	-0.0081	-0.0054	0.0201	0.0053	0.0263	0.0088	-0.0025	-0.0324	-0.0321	-0.0479
97.78	-0.0157	-0.0162	0.0088	-0.0033	0.0221	0.0111	0.0000	-0.0275	-0.0274	-0.0436
119.22	-0.0215	-0.0193	0.0068	0.0014	0.0324	0.0129	-0.0031	-0.0254	-0.0276	-0.0433
140.66	-0.0178	-0.0217	0.0072	-0.0019	0.0219	0.0058	-0.0035	-0.0264	-0.0240	-0.0382
162.11	-0.0136	-0.0100	0.0126	-0.0028	0.0192	0.0035	-0.0053	-0.0272	-0.0256	-0.0410
183.55	0.0058	0.0065	0.0246	0.0057	0.0238	0.0054	-0.0031	-0.0247	-0.0208	-0.0394
205.00	-0.0040	-0.0067	0.0053	-0.0161	0.0150	0.0008	-0.0107	-0.0308	-0.0207	-0.0460

With the exception of the mid front row, the difference matrix emphasizes the similarity of the inductive and ESP32 touch probe results. The differences cannot account for the very large compensation changes needed.

The concept of X axis twist is based on the idea that if the extruder X axis rods are twisted with respect to one another, the probe is angled with respect to a vertical axis. This produces z heights that will be larger than the true z height. Unless the rods themselves are bent relative to one another, the correction would be expected to be linear and monotoinic.

Considering the large difference in Y offset of 24 mm between the inductive probe and the ESP32 rod touch probe, the data above suggests there is little obvious X axis twist. This is somewhat in contention with the paper test data previously mentioned

The conclusion after all this extra effort is that the inductive probe is faithfully reading the bed mesh heights, as it sees it from the extruder. There is no strange heterogeneity problem of the bed material, or bed thickness problems. The distortion is real.

The first ESP32 experiments with the probe rod fixed to the back of the extruder had a very different y offset of 76 mm. The x offset was very similar at 27 mm, but the y offset was around 64 mm. Thus, nearly 1/3 of the bed could not be probed, so the data was not ideal for help in determining what the cause of the distortion was. Still, this early experimental data does have some use with respect to the whether the distortion was due to X axis twist.

and have a much greater effect on the reported z heights.

	27	**46.3**	**65.6**	**84.9**	**104.2**	**123.5**	**142.8**	**162.1**	**181.4**	**200.7**	**220**
76.33	-0.140	-0.134	-0.061	-0.061	-0.203	-0.172	-0.147	-0.132	-0.142	0.032	-0.041
96.85	-0.178	-0.154	-0.064	-0.064	-0.054	0.007	0.031	0.057	0.038	-0.008	-0.073
117.38	-0.144	-0.105	-0.020	-0.020	0.009	0.043	0.068	0.089	0.068	0.016	-0.060
137.90	-0.106	-0.084	0.032	0.032	0.013	0.055	0.087	0.117	0.095	0.051	-0.003
158.43	-0.053	-0.028	0.096	0.096	0.044	0.080	0.106	0.109	0.106	0.069	0.022
178.95	-0.002	-0.077	0.117	0.117	0.082	0.096	0.116	0.141	0.127	0.099	0.037
199.48	0.061	0.092	0.100	0.100	0.105	0.130	0.137	0.149	0.158	0.110	0.050
220.00	0.024	0.035	0.086	0.086	0.085	0.141	0.164	0.154	0.158	0.102	0.006

This correction matrix from these early ESP32 measurements further reinforces the skepticism that X axis twist is the major factor in this specific case. With the huge Y offset, even a small angular difference would lead to a large effect on the z heights. Even though there is some evidence of a pattern, the magnitude of the changes across the board do not exhibit any sort of constant magnitude, especially associated with the xy axis compensation values that were found. There could be some contribution from X axis twist, but it does not appear as the dominant problem.

ADXL345 Measurements

The adxl operation and data handling will be described, with the thought that someone else may find some sort of use for the idea, or how the data reduction was done. However, the data from all adxl tests was discarded for several reasons: After doing the runs, I realized that the bed had not been checked for flatness by Z tramming, and was tilted by several hundred micrometers. It was not possible to unambiguously define when the bed tilt happened, making any conclusions about the data suspect.

The second reason for discarding all the data was that when the adxl345 module probe is directly attached to the touch probe, the wire lead going from the adxl345 module to the Pad 7, is a bit stiff. This would create side stresses on the probe rod. In addition, the rod was turning slightly with the change in stress on the wire as the extruder moved side to side. This rotation would certainly affect the X and Y data, but did not strictly affect the Z heights. This potential problem was considered before using the probe, but the hope was the wire lead was flexible and light enough to not unduly affect the rod motion. When comparing optocoupler results to the adxl345 data, there were indications that in some bed positions, the lead wire did cause side stresses, and may have influenced the rod sliding, which affected the time measurements. All of these issues became very clear only after the switch to the optocoupler measurements. Thus, all adxl345 data was discarded.

From a data logging perspective, how fast data can be recorded with an Arduino or ESP32 mcu is a significant issue. The adxl345 probe typically used in acceleration resonance tests connected to a Raspberry PI or clone, takes data continuously at a rate of 3200 Hz or 313 microseconds/read.

Assuming that the probe is set to a travel velocity of 5 mm/s that represents ~ 0.000313 s*5 mm/s = 0.001565 mm resolution between data points. That resolution is certainly below the average error between different measurements of the same bed position over different, separate measurements, at least for my SV06. Typically, resonance testing lasts for 130 s and takes 170,000 data points.

The Arduino module was not going to read data at the fastest rate. Even getting a ESP32 to achieve a data rate of 3200 Hz or even 1600 Hz is not straightforward. Fortunately, after a couple of days reading of how to do fast logging on the ESP32, I stumbled on a Klipper function that can be used to read and store adxl345 output directly to the Raspberry Pi.

The "ACCELEROMETER_MEASURE", Klipper function allows a user to take data at various rates, and after acquiring the data, dump it to a file. Calling the function, starts data logging to memory. Calling it again, stops the process. The *.csv* file generated is then dumped to the *~/tmp/* directory, where it can be downloaded using a terminal program like WINCP. The file must be downloaded before the PI is turned off, because all files in the */tmp* folder are automatically erased. Klipper's acceleration resonance testing operation probably uses this function. The macro to do this can be found in the current folder, as *adxl_touch_test.cfg*.

As for the ADXL345 module, I decided to use the BTT version that came with the BTT Pad 7. A special adapter plate was printed to fit on the end of the probe rod, to which the 345 was taped.

The idea behind the use of the adxl was to register the time for the extruder to traverse a certain known distance based on acceleration changes. With the adxl attached directly to the probe head, there were five possible accelerations: 1. when the extruder stops traversing in the x/y direction and begins dropping, 2. when the rod probe makes first contact with the bed, 3. when the extruder begins to rise from the bed, 4. when the touch probe moves off the bed, and 5. when z motion stops, and the extruder starts moving to the next position.

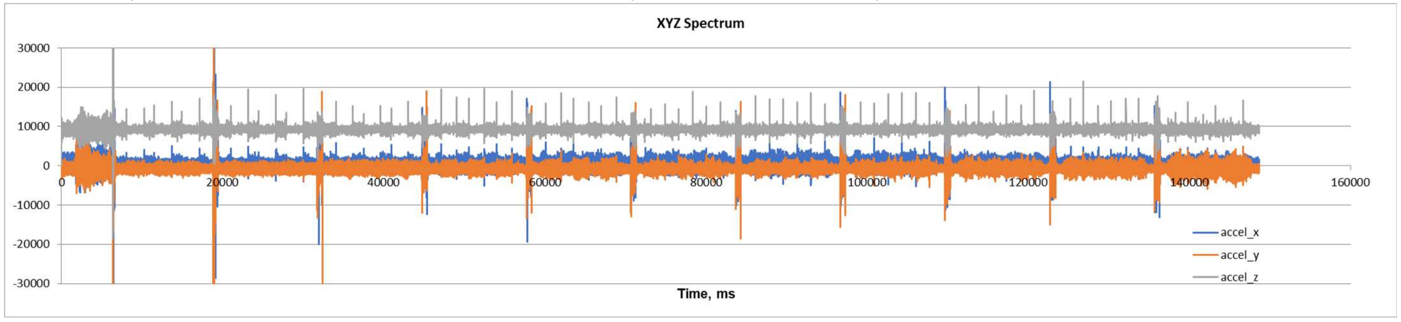
The distance to move was generally set to 2 mm, from 5 mm to 3 mm and back. This extruder distance range was not sufficient to trigger the inductive probe. G1 commands to the z axis stepper motors moved the extruder the required distance up and down. The distance of the touch rod at rest against an adjustable stopping collar on the touch rod varied, but generally was set between 1 and 2 mm above the bed surface.

Data was collected for 10 X axis positions, corresponding as much as possible to the normal 10 inductive probe X positions. Each of the 10 Y axis position rows were collected separately resulting in 10 files to analyze. The number of tests at each point varied a bit over the course of the experiments and were either 3 or 4 measurements/position.

One important point about the use of a DIY touch probe is what data is being read. Apparently, Klipper measures distances indirectly, by counting stepper motor steps, and converting to distance. I was not able to discover any macro or function, that accessed the step counting functions of Klipper. With the DIY touch probe, time is the only parameter that is available. Knowing the distance traversed and the time interval, the velocity could be calculated. There are acceleration and deceleration that Klipper automatically applies to extruder movement. Thus, it is necessary to experimentally get the information on travel velocity. A simple macro was set up, *adxl_time_test.cfg*, (see folder) that started a microsecond timer and did 40 z height cycles. From the time it took to do the cycles, the nominal velocity set at 5 mm/s was calculated as 4.93E-06 mm/ųs. This value was used in all experiments to calculate z height positions at any time.

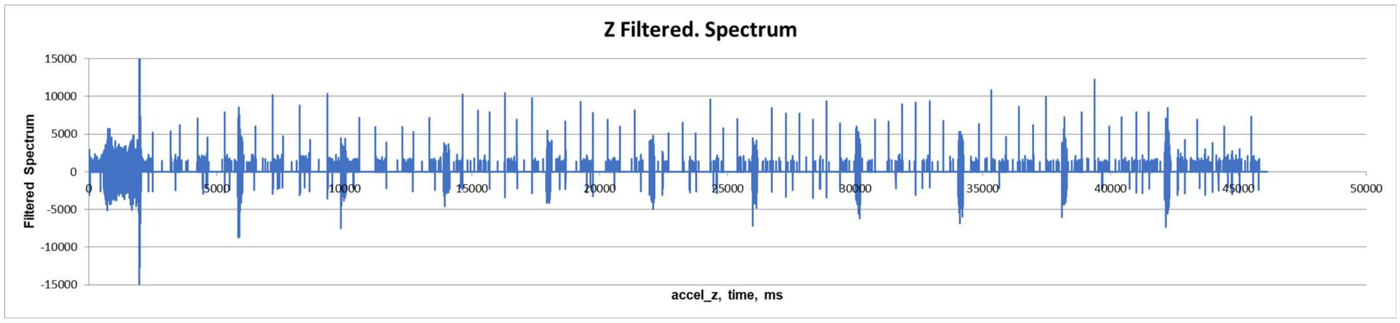
The idea was to measure the time for the touch probe to trigger when first contacting the bed, hit the bed, start rising and finally trigger when the rod collar stopped the rod movement. The data received did not look as naively obvious as described for the acceleration/deceleration points, and required some time to understand before even beginning to work with the time differences. A set of filters were needed to get to a readable analysis.

Because the way the lead wire was connected to the adxl345 probe, the Y axis of the module pointed, for the most part, in the X bed direction. Below is a plot of the raw output for all 3 adxl axes:



The highest magnitude peaks, particularly evident in the Y channel, represent the extruder traversing to the next point, and beginning descent from an initial 5 mm to 3mm z height. The desired information is buried between these limits with varying noise levels.

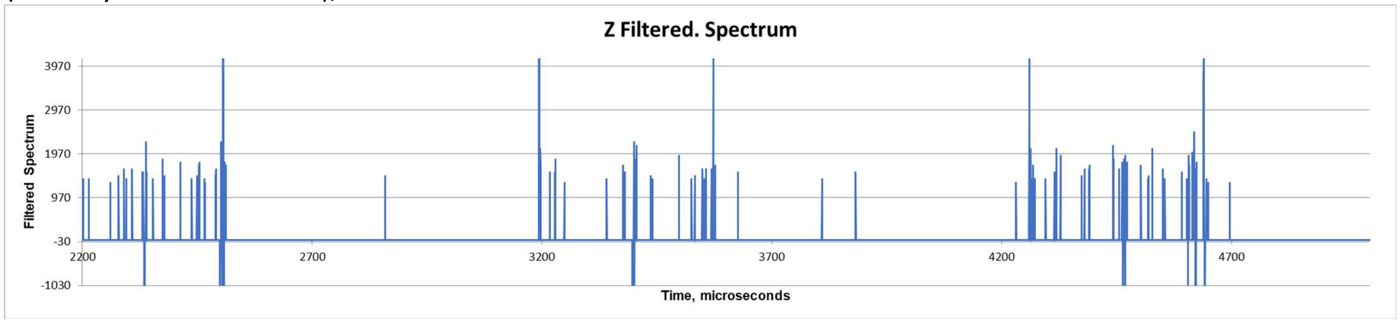
The time data was reset by referencing to time = 0 microseconds as the first entry. The first stage noise reduction filter of the z axis positions was to just take the finite first derivative of the time data; this is simply the difference between successive time values. More sophisticated derivative methods can be used, but would effectively smear out the time resolution. The same can be said for moving average filtering. The simple derivative was surprisingly good at removing noise. From the sample data above the filtered data is shown below:



The remaining noise turned out to be instructive in how the probe was acting to acceleration changes, and understanding where the real time events were positioned.

The next filtering that was done was to break out the each of the ten positions. This was done by using the Y axis module data to demarcate the start and stop of the z movement cycles at a specific xy position. Using an Excel macro on the derivative data, the Y positions were filtered by using a negative value that would capture the start of a measurement. The value was rounded to the nearest thousand, and recorded by the macro on a separate worksheet. Because the minimum Y values were a region of time values, the criteria for choosing the start position for the z probe action was the last low value above twice the standard deviation of the noise.

Each of these position time ranges needed to be independently expanded. Initially, this was manually done, but it immediately became clear that it would take days to go through the data for 10 x 10 probes positions to extract the relevant time data. A worksheet ListBox was created from the rounded Y values. User clicking on a listbox range in the worksheet signified a change in the ListBox choice. A second worksheet macro isolated the data from the derivative data and plotted it. Below is an example of an expanded section (actually a “sub-section”),



Initially, it was not clear where the critical data was between the Y traverses. Several background tests where the rod trigger height and the extruder height interval were adjusted to find out where the critical data was.

The regions between higher points, for example between ~3200-3600 ųs, is the time the extruder was moving up from where the touch probe stop collar had locked the probe in the up position as the extruder continued moving up. At ~3400 ųs, the extruder had reached the upper height setting, and started down again; at ~3600 ųs, the rod probe first touches the bed. This is the start of the low noise region, where the touch rod tip is sitting on the bed, as the extruder continues down to the lowest height dictated by minimum height setting defined in the Klipper macro code. The point at ~4300 ųs represents where the stop collar locked the rod from moving, generating the final acceleration spike for that measurement cycle.

The higher noise level region is believed to be due to the free vibration of the rod motion being detected by the accelerometer. The less noisy region, is where the probe is contacting the bed, and represents damping by the soft coating of the bed. About half way into the low noise regions, can be seen a single spike. This is the reverse acceleration when the extruder bottomed out at the requested minimum height and began to rise again. This point did not always show up in every cycle at every bed position. The reason was probably due to the aforementioned issue with wire lead stress on the touch rod. The time for a height measurement was taken as one half the time of the low noise region; this time was then converted to a height measurement.

Conclusion

The bottom line on what causes the particular SV06 bed mesh problems remains elusive, despite a number of different tests. Certainly, the inductive probe is not at fault, evidenced by the very similar ESP32 data.

X axis twist does not appear to be a major player unless we invoke some sort of complex nonlinear distortion of the extruder path in the X axis. It might be possible to use the x axis twist function, but with many more paper test points to do, and most likely the incorporation of some sort of nonlinear predictive equation. Certainly, the nonlinearity of the problems does suggest why the x axis twist tests fail. In addition, it is possible that the constant issue that the paper test to get z offsets, has a chronic issue with the printer will affect the twist results. The values always must be increased manually by 0.02-0.04 mm to properly start any print sticking to the bed.