

## Communication between User (Prompt) and ChatGPT (Response)

### Prompt 1

I want to use an FDM printer with a 0.2 mm nozzle and PEL filament to print microfluidic devices. Recommend reasonable printing settings and geometric parameters like minimum wall thickness, channel diameter, and layer height that I can use for initial review.

### Response 1 (ChatGPT)

For a 0.2 mm nozzle with PEL, a good starting point is:

- Layer height: 0.08–0.12 mm
- Print speed: 15–25 mm/s for fine features
- 100% infill around channels

These values give a balance between resolution and printability for microfluidic-like features.

### Prompt 2

Based on this, suggest specific default design parameters for my microfluidic channels so I can use them in code. I want spiral, T-junction, and Y-junction devices.

### Response 2

You can use:

- $w = 300 \mu\text{m}$  (outer diameter)
- wall ratio = 0.30 (gives 90  $\mu\text{m}$  wall and 120  $\mu\text{m}$  lumen)
- Spiral:  $r_0 = 200 \mu\text{m}$ ,  $s = 2000 \mu\text{m}$ , loops = 3, 4 or 5
- T/Y:  $h = 2000 \mu\text{m}$  (main length),  $v = 1500 \mu\text{m}$  (branch length), angle = 45° for Y.

These parameters are consistent with your FDM constraints and allow hollow channels with printable walls.

### Prompt 3

Now using these parameters, write a Python script that communicate with API and generates spiral, T-junction, and Y-junction devices and exports STL files for FDM printing.

### Response 3

I created a Python script using trimesh and numpy that:

- Defines default parameters ( $w$ ,  $r_0$ ,  $s$ , loops,  $h$ ,  $v$ , angle, wall ratio)
- Builds three devices: spiral, T-, and Y-junction
- Uses simple solid extrusions for channels
- Exports one STL per device into an output folder.

- Effectively communicates with API.

#### **Prompt 4**

I need the channels to be hollow circular tubes, not solid blocks. The lumen must stay open for flow.

#### **Response 4**

I updated the script to:

- Define a circular cross-section
- Sweep it along a 3D centreline to form **hollow tubes**
- Use outer dia = w and wall = w \* wall ratio
- Compute inner diameter as D\_inner = outer\_d - 2\*wall

Each device now has a hollow lumen rather than a solid channel.

#### **Prompt 5**

Add functions that generate the centreline for each device: a spiral (Archimedean), a proper T, and a symmetric Y.

#### **Response 5**

I added:

- build\_spiral(p): Archimedean spiral  $r(\theta) = r_0 + s\theta/(2\pi)$  in  $\mu\text{m}$
- build\_tjunction(p): straight main channel and one vertical branch
- build\_yjunction(p): one inlet and two symmetric branches at  $\pm\text{angle}$

Each function returns centerline points in  $\mu\text{m}$ , which are then converted to mm for the sweep.

#### **Prompt 6**

The T-junction looks wrong. It has extra segments and doesn't look like a clean T. Fix it so there's only one vertical branch at the centre of the main channel.

#### **Response 6**

I corrected build\_tjunction(p):

- Main channel: from  $[-h/2, 0, 0]$  to  $[+h/2, 0, 0]$
- Single branch: from  $[0, 0, 0]$  to  $[0, v, 0]$

There are no extra arms now. The sweep uses these two segments combined into one tube network.

#### **Prompt 7**

The Y-junction branches look slightly uneven. Make them exactly symmetric at 45°.

### **Response 7**

I fixed the trigonometry in build\_yjunction(p):

- Inlet: from [-h/2, 0, 0] to [0, 0, 0]
- Outlets: v length at +angle and -angle (converted to radians) from [0, 0, 0]

Now both branches are symmetric with respect to the inlet axis.

### **Prompt 8**

When I look at the spiral STL, the tube seems to twist along the curve. Can you fix the sweep, so it doesn't twist?

### **Response 8**

The twisting comes from using a naive orientation. I implemented a **parallel-transport frame** that:

- Computes tangent vectors along the centreline
- Propagates normal and binormal vectors smoothly

This reduces unwanted rotation of the circular cross-section along the spiral.

### **Prompt 9**

All STL files are not watertight. Mesh mixer shows holes in curved areas. Can you repair this?

### **Response 9**

I added a post-processing step:

- mesh.fix\_normals()
- If not watertight: fill holes, remove duplicate faces, remove unreferenced vertices
- Re-check mesh.is\_watertight and give a warning if still problematic

This significantly improves STL quality for FDM printing.

### **Prompt 10**

I want to thicken the outer walls by +0.2 mm for mechanical strength, but I do not want to change the inner lumen. Add that feature.

### **Response 10**

I added OUTWARD\_THICKEN\_MM = 0.2 and a helper outward\_chicken\_params(outer\_d\_mm, wall\_mm, delta\_mm) that:

- Increases outer diameter by 2\*delta
- Increases wall thickness by delta

- Keeps the inner diameter (lumen) unchanged

All devices now use these thickened values when building the tubes.

### **Prompt 11**

Please add functions to export 2D DXF and SVG top views of the devices using the same centrelines.

### **Response 11**

I added:

- `export_dxf_multi(kind, paths_um, p, base)`
- `export_svg_local_multi(kind, paths_um, p, base)`

These project the centerlines onto the XY-plane in  $\mu\text{m}$  and draw polylines. SVG and DXF files are saved with names based on the device type.

### **Prompt 12**

Some of my SVGs look stretched or squashed. I think  $\mu\text{m}$  coordinates are not mapped properly to the SVG view box. Fix this so they appear correctly scaled.

### **Response 12**

The deformation happened because the raw coordinates were used without computing the bounding box. I updated the SVG code to:

- Find min/max X and Y
- Add a margin
- Set the SVG width/height and viewBox from those bounds

Now the shapes are correctly scaled and not distorted.

### **Prompt 13**

Add labels in the SVG with the device name and main parameters like w, wall ratio, and loops or angle.

### **Response 13**

I modified `export_svg_local_multi()` to add a text annotation at the top center containing:

- Device type (Spiral, T, Y)
- w, wall\_ratio
- For spiral: r0, s, loops
- For T/Y: h, v, angle

The text uses a fixed font size and is placed above the geometry.

### **Prompt 14**

Sometimes the annotation text is missing or overlaps the geometry in the SVG. Why is that, and can you fix it?

### **Response 14**

It was caused by placing the text at coordinates that were outside or inside the device region. I now:

- Compute the bounding box
- Place the text a fixed distance above min\_y

This ensures the annotation is always visible and not overlapping.

### **Prompt 15**

I have noticed that the script is only using API to generate text not actual images and geometry. Now I want to generate SVGs using the OpenAI API through Python. Add a function that sends a prompt with the polyline coordinates and gets back SVG.

### **Response 15**

I added `export_svg_gpt_multi(kind, paths_um, p, base)` which:

- Builds a text description of the polylines and desired style
- Sends it as a prompt to the OpenAI API using the Python client
- Extracts the SVG XML from the response and saves \*\_gpt.svg.

### **Prompt**

If the API fails or returns nothing, my script crashes or I get an empty SVG. Make it robust and add a local fallback but make API as primary generator and if API fails and local generation occurs, mention in the python window that API failed to respond moving to local generation.

### **Response 16**

I wrapped the API call in try/except and added logic so that:

- If the call fails, times out, or the SVG is empty
- The script prints a warning and calls `export_svg_local_multi()` instead

This keeps the workflow stable even when the API is not responding, keeping your other conditions satisfied.

### **Prompt 17**

I want to log every prompt I send to the API for reproducibility. Add logging to a text file.

### **Response 17**

I created `PROMPT_LOG.txt`. Each API call:

- Appends the timestamp
- The device type
- The exact prompt string sent

This lets you reconstruct how each GPT SVG was generated.

### **Prompt 18**

I noticed that with some values of width and wall ratio, the lumen diameter becomes unrealistic or even negative. Add checks so this cannot happen.

### **Response 18**

I added parameter validation:

- `wall_ratio` must be between 0.1 and 0.4
- If  $2 * \text{wall} \geq \text{outer\_d}$ , the script raises an error and prints a message

This prevents invalid lumen geometries.

### **Prompt 19**

I want a summary of the lumen diameter calculation in the code comments so I don't forget how 120  $\mu\text{m}$  or 160  $\mu\text{m}$  were obtained.

### **Response 19**

I added comments explaining:

- $\text{wall} = w * \text{wall\_ratio}$
- $D_{\text{inner}} = w - 2 * \text{wall}$

For example,  $w = 300 \mu\text{m}$ ,  $\text{wall\_ratio} = 0.3 \rightarrow \text{wall} = 90 \mu\text{m} \rightarrow D_{\text{inner}} = 120 \mu\text{m}$ .

### **Prompt 20**

Sometimes the generated spiral path is too long or self-intersects. Add a simple check to avoid extreme parameter combinations.

### **Response 20**

I added a check on the total number of turns and final radius. If the spiral radius or total length exceed a reasonable threshold, the script warns and skips that parameter set.

### **Prompt 21**

I'd like to log each run (device type, time taken, output files, STL hash) into a CSV file. Can you add that?

### **Response 21**

I added `device_runs.csv`. Each device build records:

- A random session ID
- Device kind
- Runtime in seconds
- STL filename
- MD5 hash of the STL
- Paths to DXF and SVG files.

### **Prompt 22**

Some DXF files do not open correctly in my CAD viewer. Can you ensure the layer and scale are simple and compatible?

### **Response 22**

I simplified the DXF export:

- Single layer name
- Coordinates kept in  $\mu\text{m}$  units
- Polylines written with standard DXF entities

This improves compatibility with most viewers.

### **Prompt 23**

I see occasional “list index out of range” errors in the sweep function for very short paths. Fix this.

### **Response 23**

The error occurred when paths had fewer than 2 points. I added guards:

- If the path length is too short, the script prints an error and skips building that device.

### **Prompt 24**

The mesh normals sometimes look flipped. Fix them systematically.

### **Response 24**

After mesh construction, I now always call `mesh.fix_normals()` to orient faces consistently outward.

### **Prompt 25**

When I use a 0.2 mm nozzle, I worry about too-thin walls. Add warnings if the wall thickness is < 0.2 mm in the final model.

**Response 25**

I added a check: if wall thickness (in mm) is less than 0.2, the script prints a warning that this may be below safe nozzle resolution.

**Prompt 26**

Sometimes the API-generated SVG cannot be opened. Can you detect that?

**Response 26**

I added a basic validation: after receiving the response, the script checks that:

- The content contains <svg and </svg> tags

If not, it treats the result as invalid and falls back to the local SVG method.

**Prompt 27**

Add a summary after everything runs, so I know which files were created.

**Response 27**

At the end of main(), the script now prints a summary list for each device:

- STL, DXF, SVG, GPT-SVG filenames
- Whether STL is watertight
- Any warnings encountered.

**Prompt 28**

I want to easily change parameters like w, loops, and angle without editing code all over. Centralize them.

**Response 28**

I consolidated parameters into a DEFAULT dictionary and a simple small config structure. The builders now read from this single location.

**Prompt 29**

I noticed that the Y-junction sometimes slightly deviates when I change angle. Can you confirm the geometry remains symmetric?

**Response 29**

I checked and corrected the vector math so that for any chosen angle, the two Y branches are truly symmetric about the inlet axis.

**Prompt 30**

For documentation, I want a CSV with centreline coordinates of each device. Add an option to save that.

**Response 30**

I added a simple CSV export that writes the (x, y, z) centerline points in  $\mu\text{m}$  for each device.

**Prompt 31**

Some of my SVGs used very thin lines that are hard to see. Increase the stroke width in the local SVG generator.

**Response 31**

I increased the stroke width proportionally to the thickened outer diameter so that channels are clearly visible in the 2D drawings.

**Prompt 32**

I need the output folder to be consistent and separate for each parameter set. Can we name folders accordingly?

**Response 32**

Yes, I set the output folder name based on key parameters (e.g., fdm\_outward\_w300\_set1) so runs with different parameter sets go into different directories.

**Prompt 33**

During some runs, the API was slow, and the script felt blocked. Add a timeout for the API call.

**Response 33**

I added a timeout parameter to the API client request. If the response takes too long, the code aborts the GPT call and retries from the start.

**Prompt 34**

Sometimes the spiral channel becomes too long and the device doesn't fit on my printer bed. Add a check or comment about that.

**Response 34**

I added a check on the final spiral radius and path length. If they exceed a simple threshold (e.g., bed radius), the script prints a warning suggesting parameter reduction.

**Prompt 35**

The code is starting to get long. Make it easy to read and modify.

**Response 35**

I refactored main() to:

- Loop through device types
- Use shared build and export routines
- Keep only minimal logic in the main control flow.

### **Prompt 36**

I want the STL filenames to be short but still indicate device type and parameter set.

### **Response 36**

I renamed them to:

- spiral\_model.stl
- tjunction\_model.stl
- yjunction\_model.stl

and keep detailed metadata in device\_runs.csv and SVG annotations instead.

### **Prompt 37**

Add a short docstring or header that explains the overall purpose of this script in one paragraph.

### **Response 37**

I added a header comment stating that the script:

- Generates parametric spiral, T-, and Y-junction microfluidic devices,
- Produces hollow channels for FDM printing with a 0.2 mm nozzle in PEL,
- Exports STL, DXF, and SVG (local and GPT-based) files,
- Logs runs and parameters for reproducibility.

### **Prompt 38**

When changing wall ratio slightly, the device performance changes. Can we highlight somewhere that the lumen size is sensitive to this parameter?

### **Response 38**

I added a comment near the parameter definition explaining that:

- Small changes in wall\_ratio significantly affect lumen diameter,
- This directly influences flow resistance and printability,  
and users should adjust it cautiously.

### **Prompt 39**

Sometimes the spiral path touches itself when loops are large. Add a simple warning when curvature becomes too tight.

### **Response 39**

I added a basic curvature check and a warning if segment angles exceed a certain threshold, suggesting fewer loops or larger pitch.

**Prompt 40**

Please add clear print statements when each device build starts and ends.

**Response 40**

I inserted logging messages like “Building spiral...” and “Finished spiral (watertight = True/False)” to make the console output easier to follow.

**Prompt 41**

Check that DXF and SVG filenames are consistent and share the same base name as the STL.

**Response 41**

I standardized naming so that, for example, spiral\_model.stl, spiral\_model.dxf, spiral\_model.svg, and spiral\_model\_gpt.svg all share the same base.

**Prompt 42**

Sometimes I forget which prompt generated which version. Can you suggest adding a version string or script revision date?

**Response 42**

I added a simple SCRIPT\_VERSION and date in the header comments and printed them when the script runs, so you can track which version produced the outputs.

**Prompt 43**

I want to ensure that the script is usable on another machine without the API key set. Confirm the behaviour in that case.

**Response 43**

If OPENAI\_API\_KEY is not found, API\_OK is set to False and all GPT-dependent functionality is automatically disabled. The script still generates STL, DXF, and local SVG files normally.

**Prompt 44**

Add a note in the comments that the script was co-developed with an LLM and that the LLM is not listed as an author.

**Response 44**

I added a brief comment:

“This script was iteratively developed with assistance from a large language model (ChatGPT). The model is used as a tool and is not considered an author.”

**Prompt 45**

Is there any final thing I should watch out for when changing parameters for new device designs?

### **Response 45**

I noted the main risks:

- Setting wall\_ratio too high or too low → unprintable walls
- Very small lumen diameters (< 100 µm) → clogging or incomplete printing
- Very long spirals → exceeding printer bed size

Users should change one parameter at a time and visually inspect STL/SVG output.

### **Prompt 46**

Give me the final, cleaned-up version of the script with all these features and fixes included.

### **Response 46**

I provided the complete final script with:

- FDM/PEL-friendly defaults
- Hollow circular channels
- Spiral, T-, and Y-junction geometries
- Outward wall thickening
- STL + DXF + SVG (local + GPT) export
- Prompt logging, CSV logging, parameter validation, and explanatory comments.