Wood instead of ivory

Functional programs for CNC machines

David Luposchainsky (aka quchen)
TNG Technology Consulting

Part 1: Overview

What is a line?

```
<svg width="200" height="200" xmlns="http://www.w3.org/2000/svg">
    <path d="M10 10 L 500 100" stroke="black" stroke-width="5"/>
</svg>
```

What is a line?

- Abstract idea with representation?
- Abstract idea without representation?
- Pixels?
- Vectors?

My lines

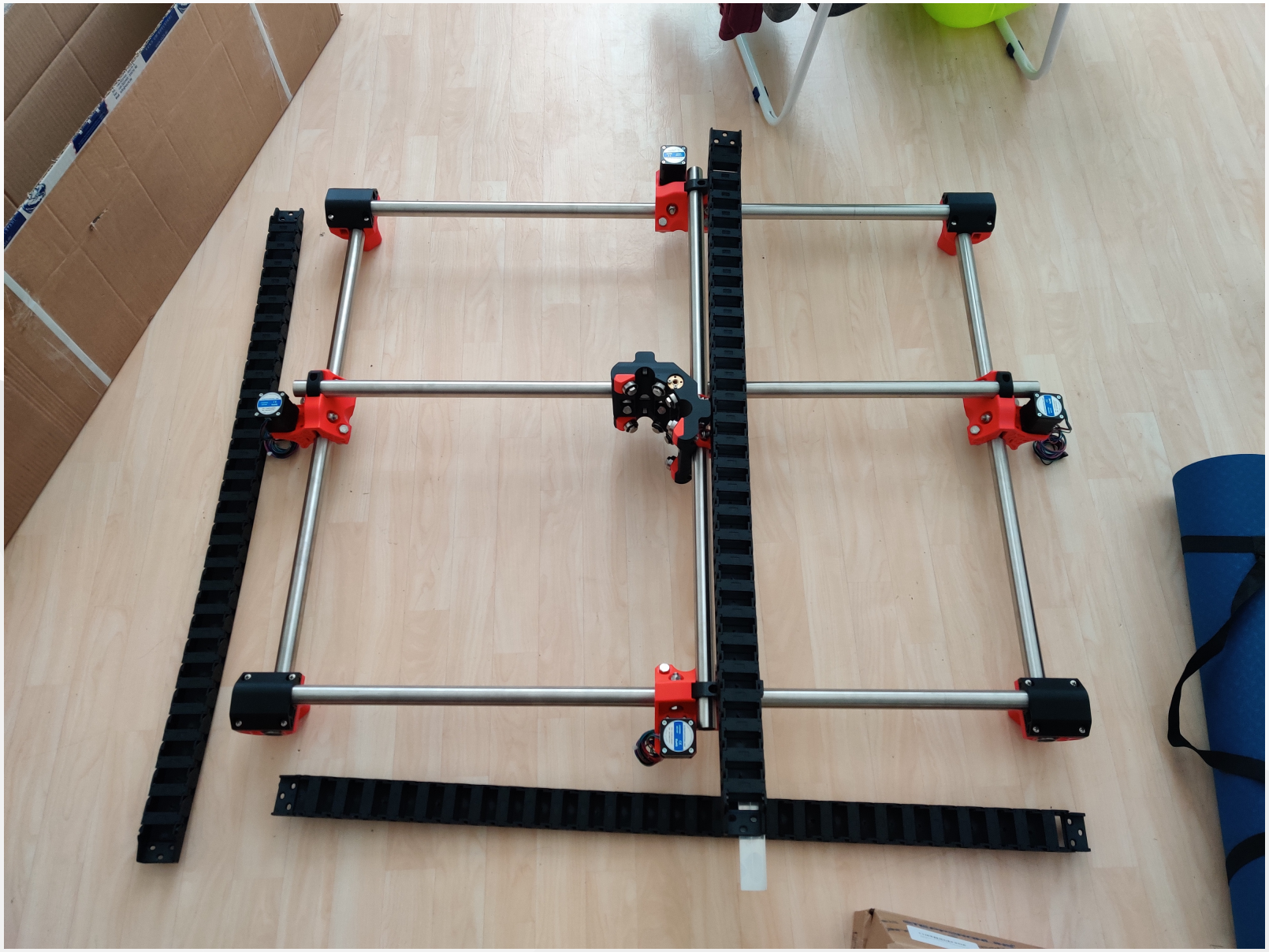
My lines are physical movement of a machine.

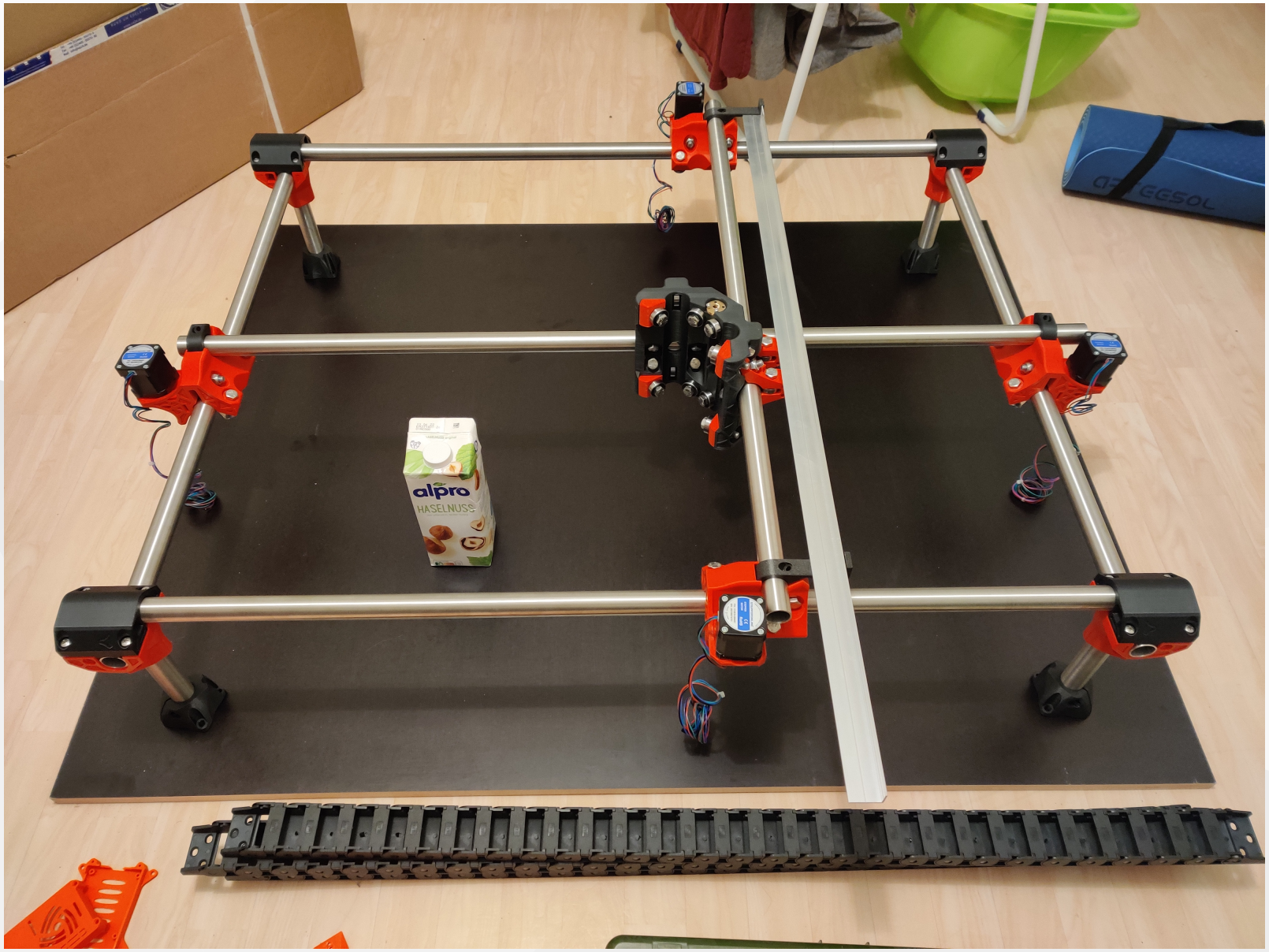
Video time! :-)

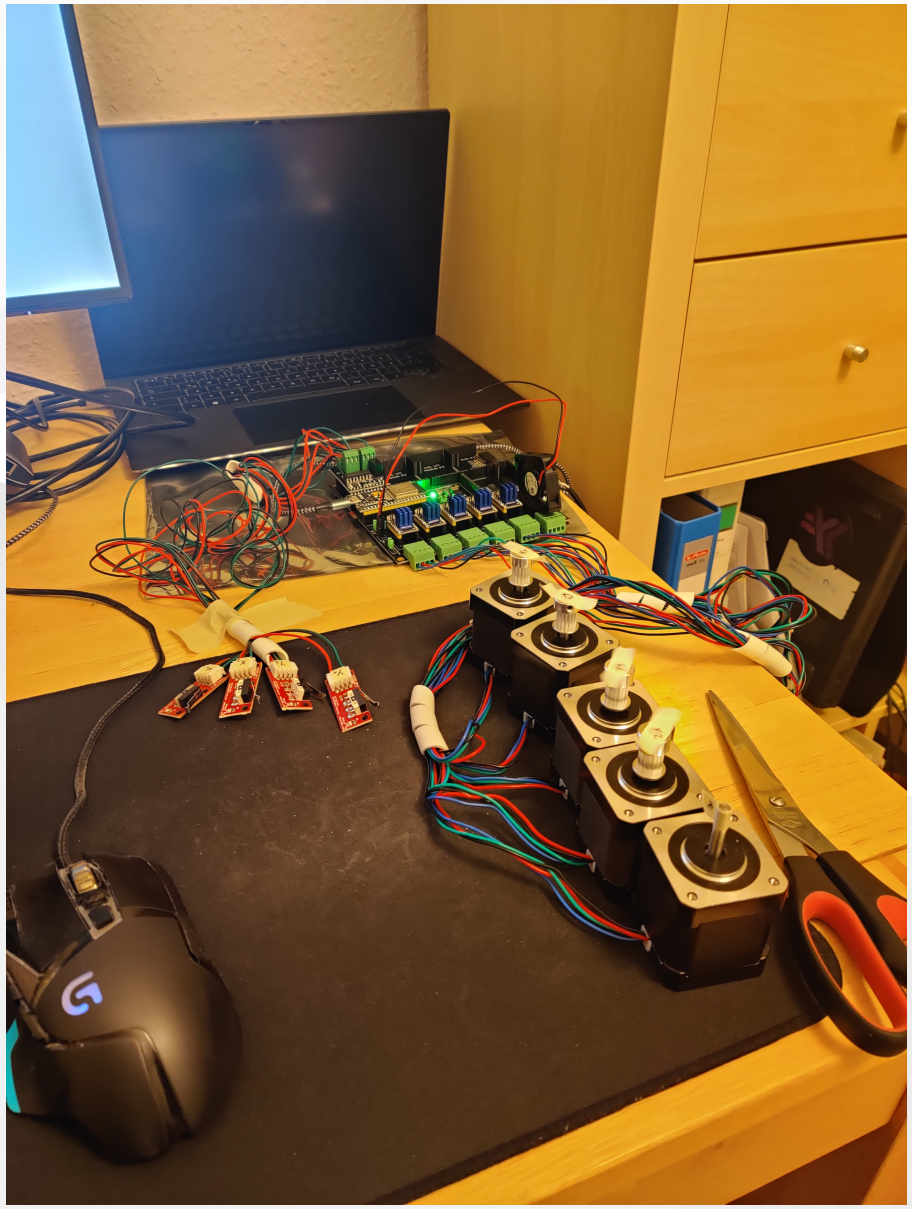
Why?

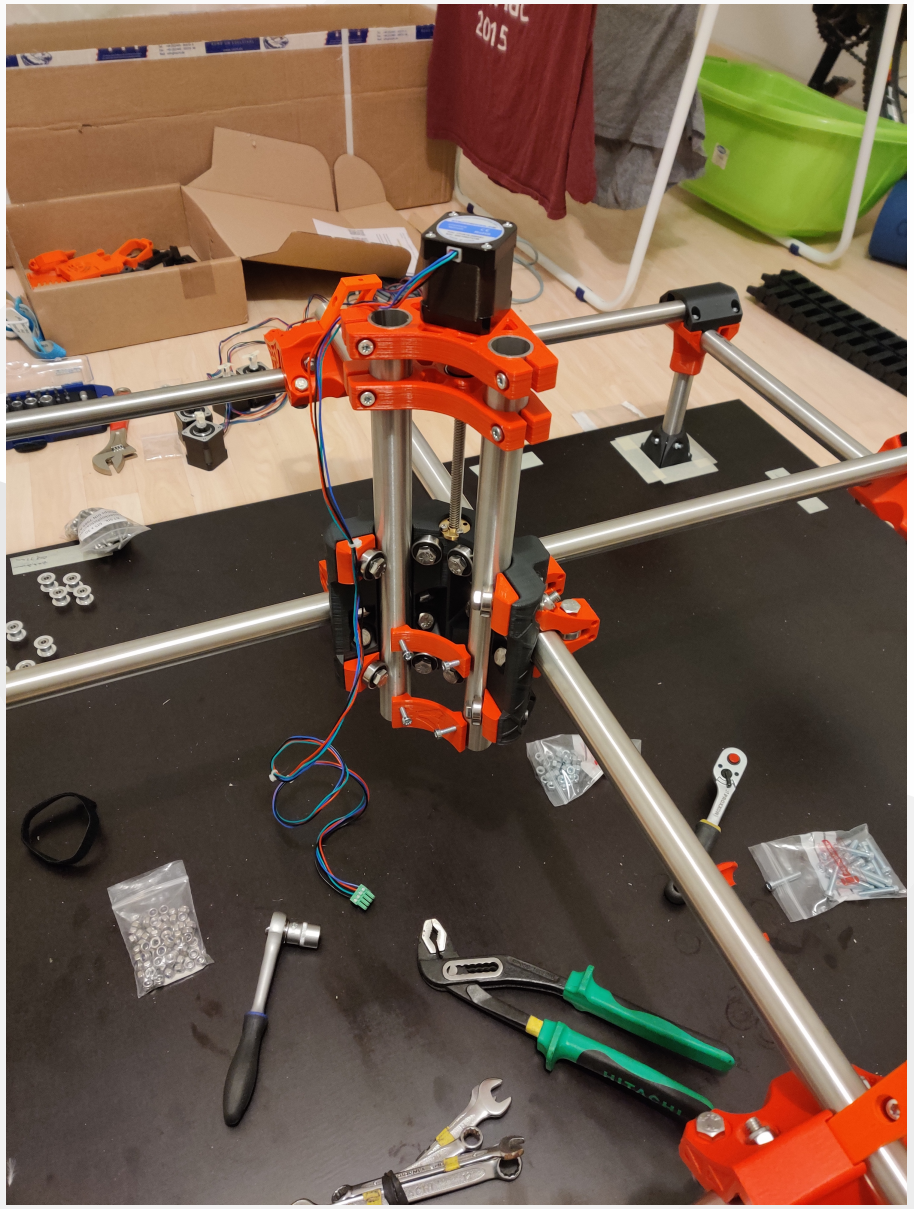
- Art
- Mechanical engineering
- Programming

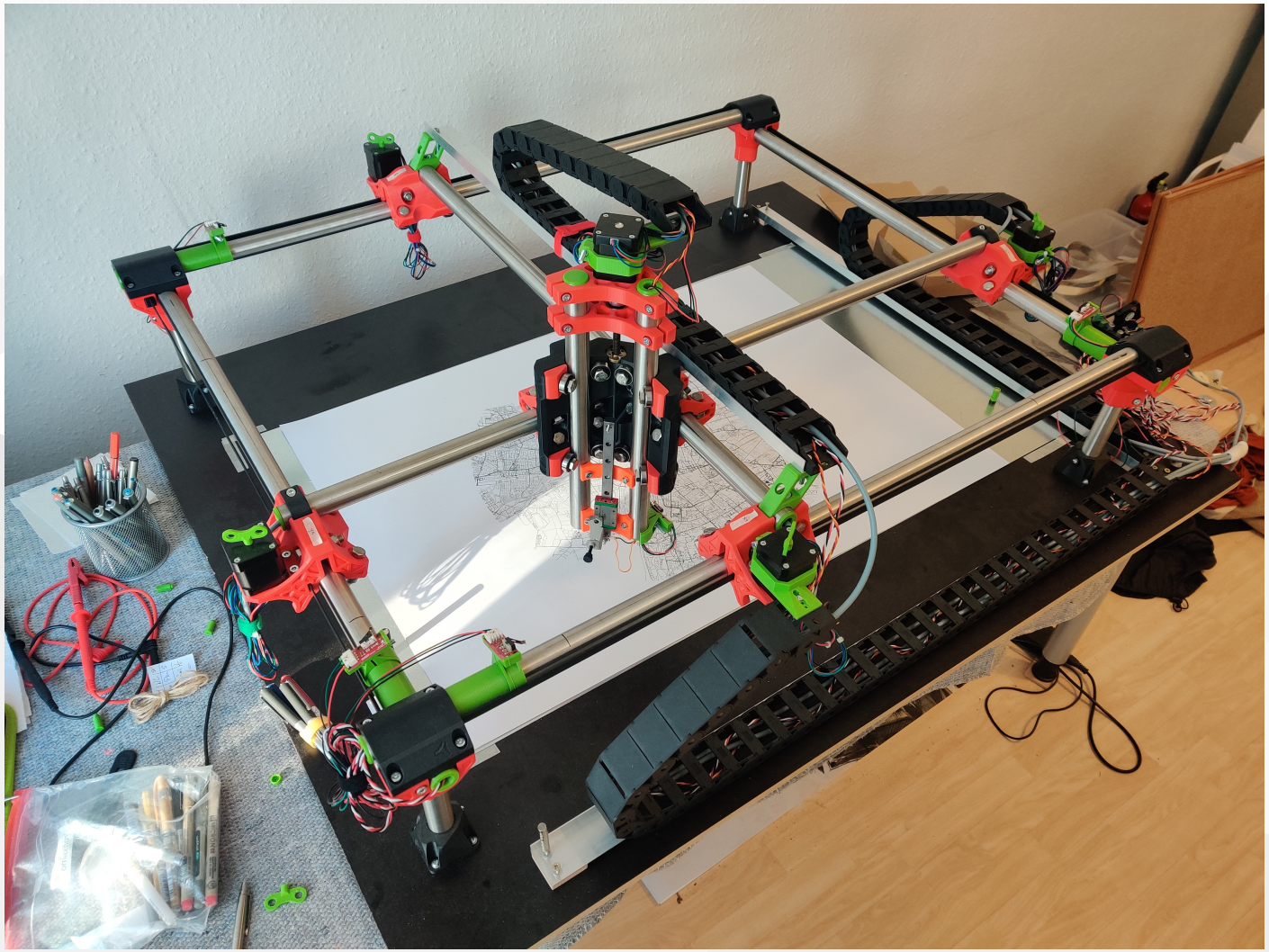
Building montage




alpro
HASELNUSS







Why art?

Hold your code in your hands.

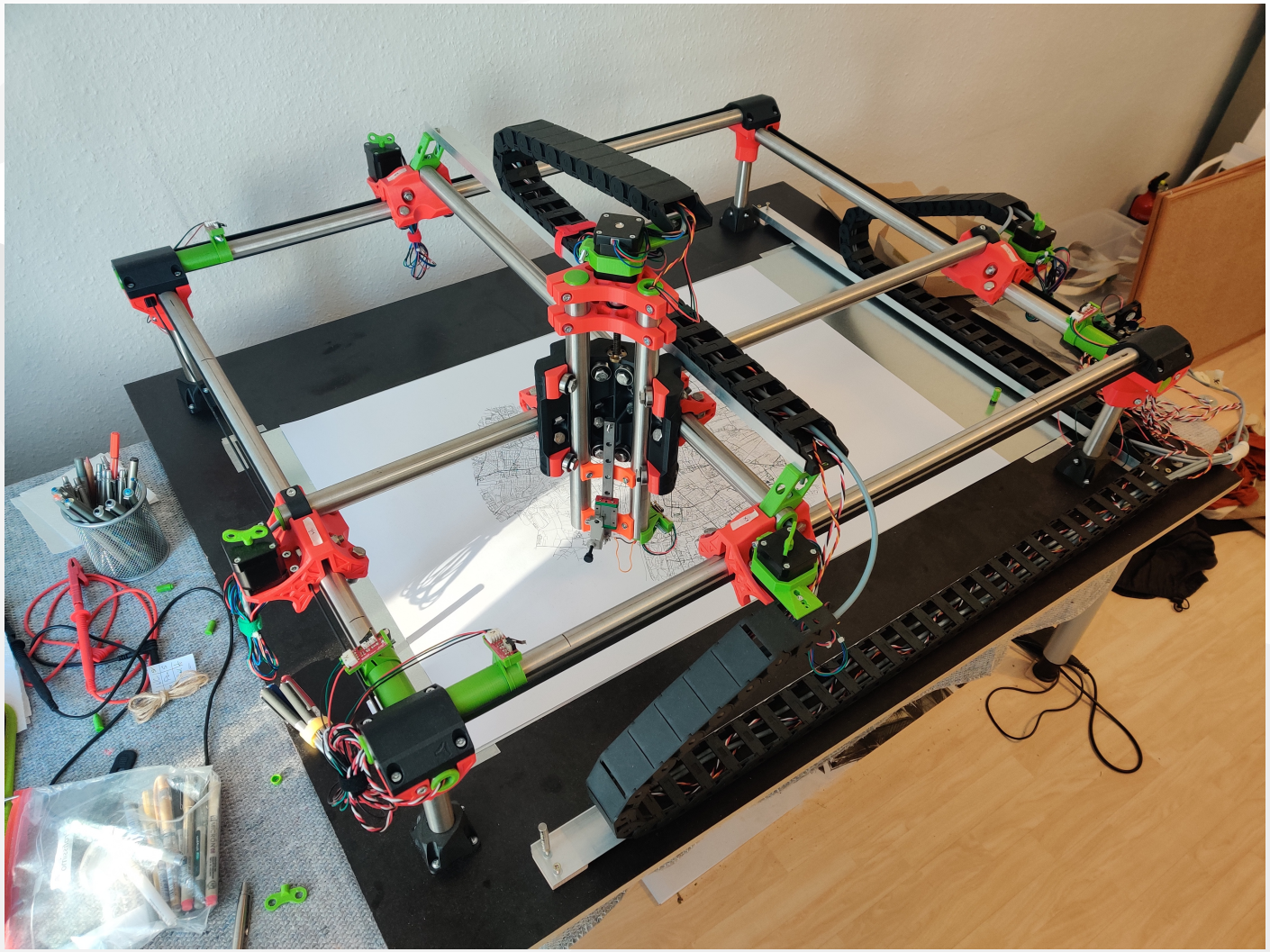
See other’s reactions.

Run out of wall space at home.

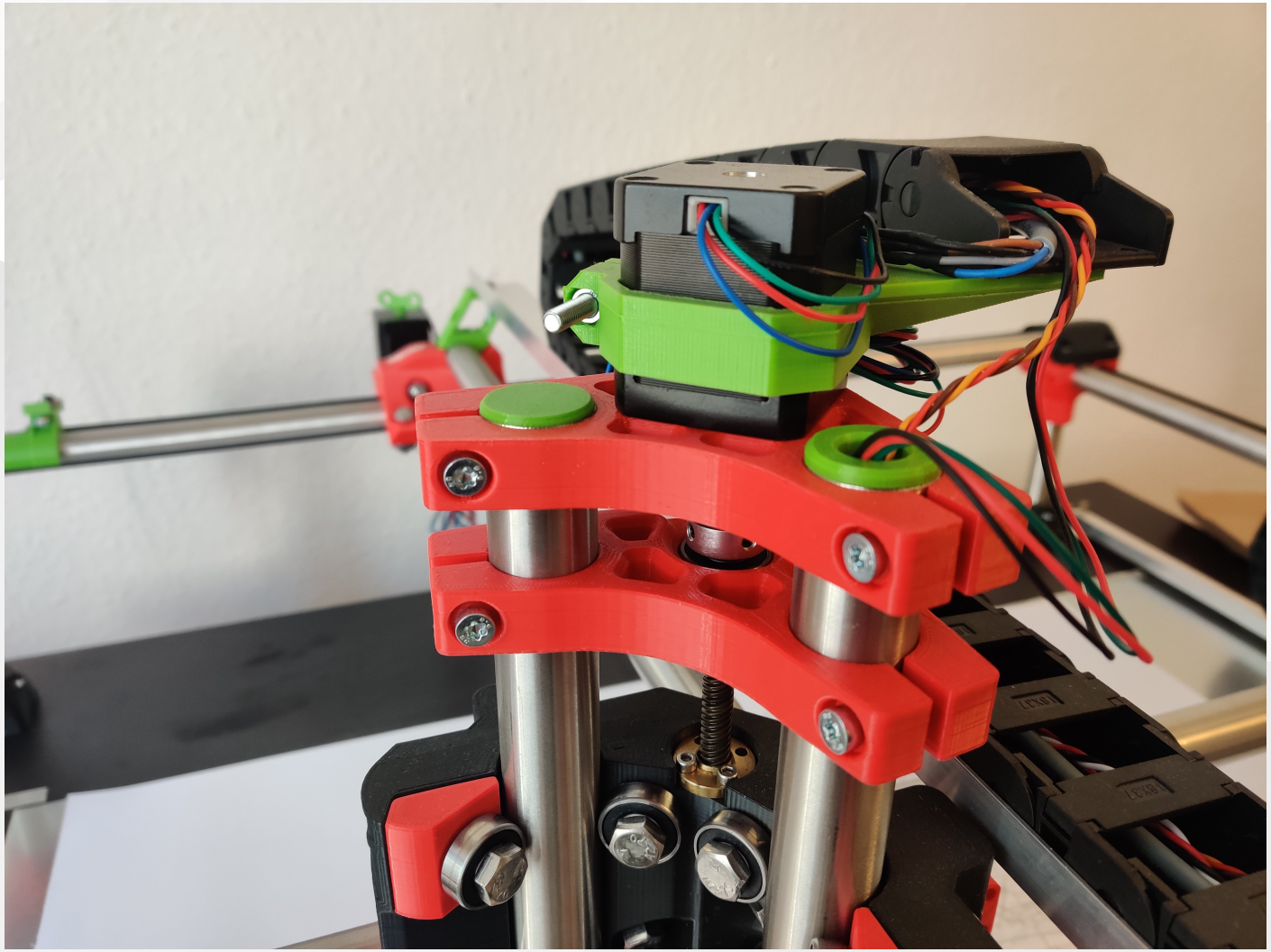
Part 2: How a CNC works

1. Hardware
2. Firmware
3. Input

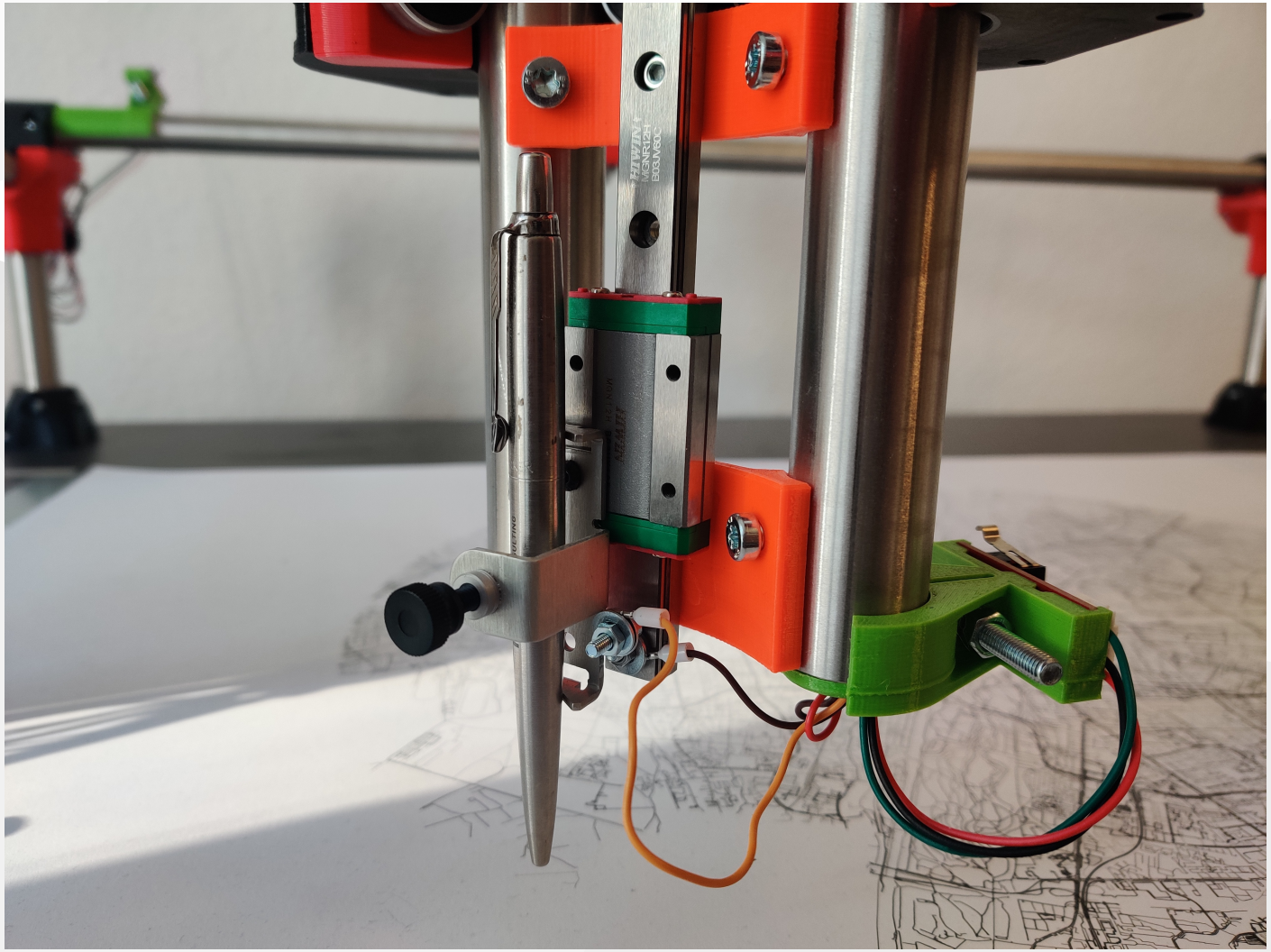
Hardware



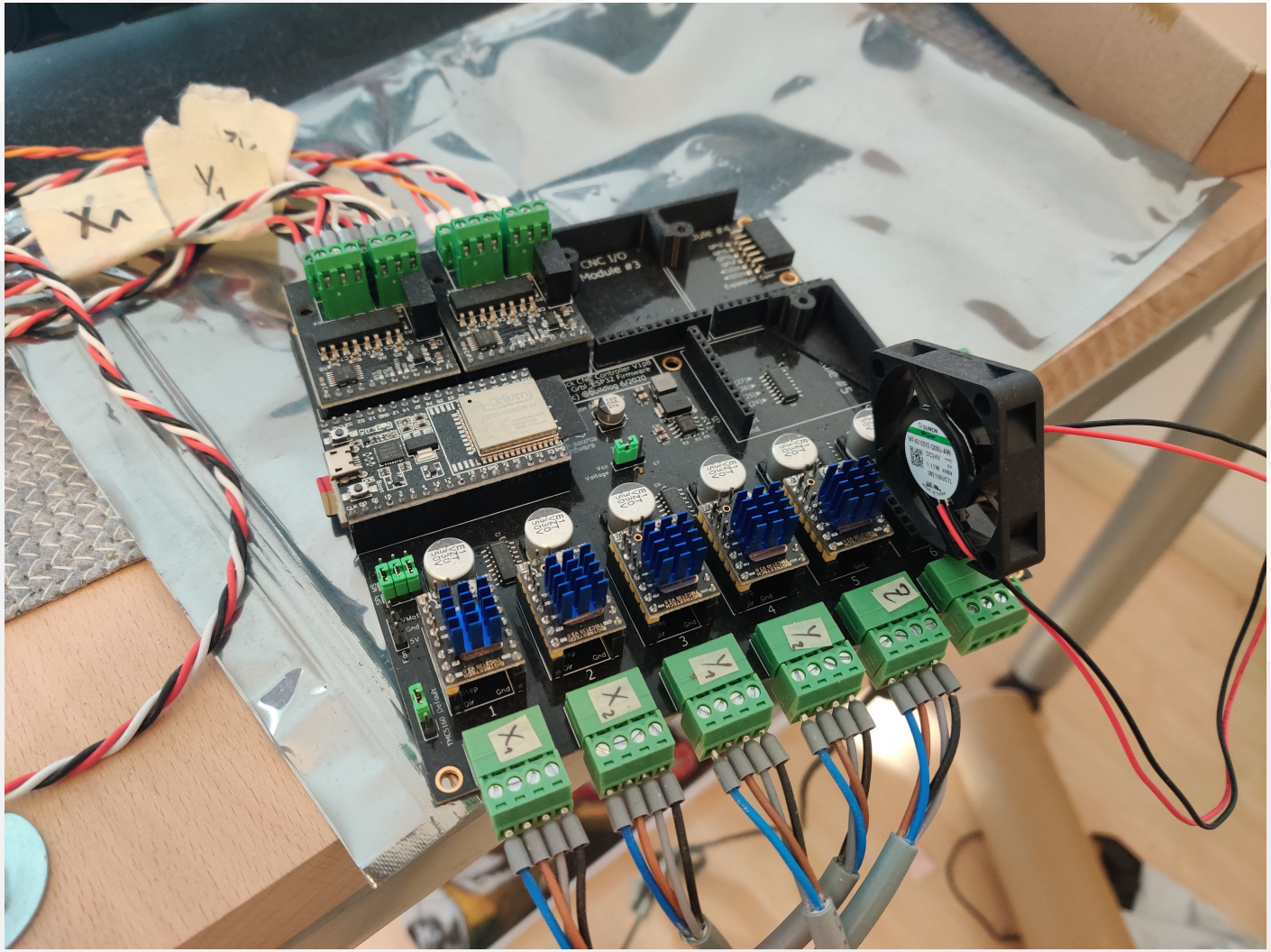
Motors



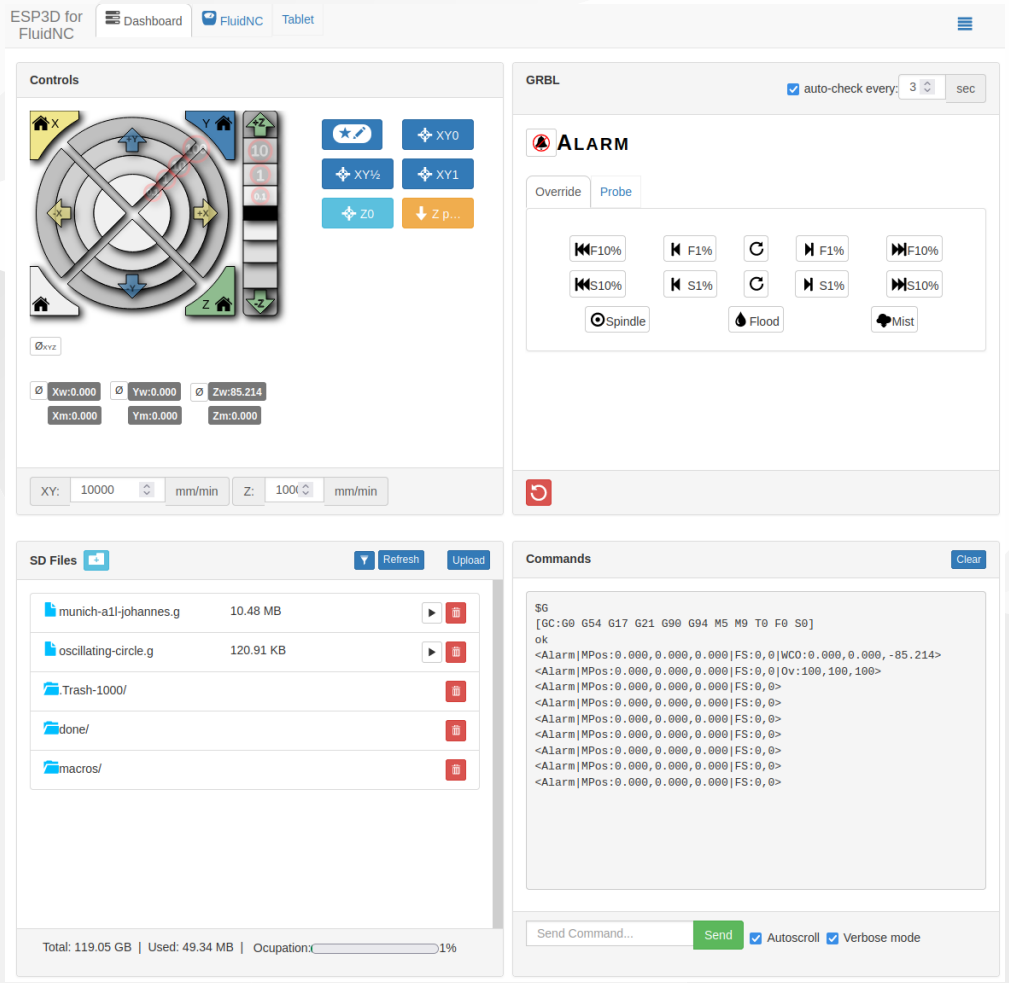
Pen holder, Z axis



Circuit board



Firmware


ESP3D for FluidNC
Dashboard
FluidNC
Tablet
Controls
XY0
XY½
XY1
Z0
Z p...
Xw:0.000
Yw:0.000
Zw:85.214
Xm:0.000
Ym:0.000
Zm:0.000
XY:
10000
mm/min
Z:
100
mm/min
GRBL
auto-check every:
3
sec
ALARM
Override
Probe
F10%
F1%
F1%
F10%
S10%
S1%
S1%
S10%
Spindle
Flood
Mist
SD Files
Refresh
Upload
munich-a1l-johannes.g
10.48 MB
oscillating-circle.g
120.91 KB
.Trash-1000/
done/
macros/
Total: 119.05 GB | Used: 49.34 MB | Ocupation:
1%
Commands
Clear
$G
[GC:G0 G54 G17 G21 G90 G94 M5 M9 T0 F0 S0]
ok
<Alarm|MPos:0.000,0.000,0.000|FS:0,0|WCO:0.000,0.000,-85.214>
<Alarm|MPos:0.000,0.000,0.000|FS:0,0|Ov:100,100,100>
<Alarm|MPos:0.000,0.000,0.000|FS:0,0>
<Alarm|MPos:0.000,0.000,0.000|FS:0,0>
<Alarm|MPos:0.000,0.000,0.000|FS:0,0>
<Alarm|MPos:0.000,0.000,0.000|FS:0,0>
<Alarm|MPos:0.000,0.000,0.000|FS:0,0>
<Alarm|MPos:0.000,0.000,0.000|FS:0,0>
<Alarm|MPos:0.000,0.000,0.000|FS:0,0>
Send Command...
Send
Autoscroll
Verbose mode

Part 3: Coding for CNC

1. Idea
2. Code digital graphic
3. Render SVG/PNG to iterate
4. Convert graphic to *lots* of lines
5. Convert to GCode
6. Calibrate plotter
7. Plot!

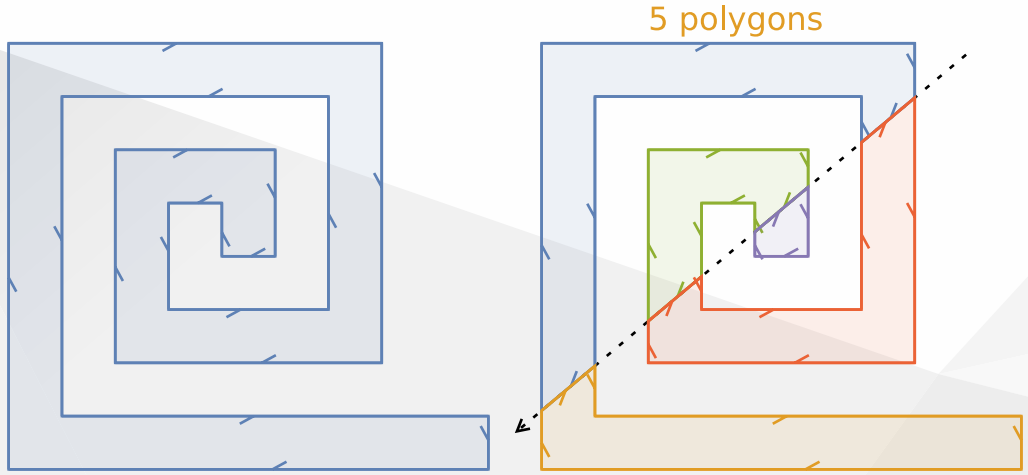
Lines

```
Haskell: Line (Vec2 0 0) (Vec2 100 10)
Python:  Line(Point(0,0), Point(100,10))

==>

"""
G0 Z10                  # Raise pen
G0 X0 Y0                # Go to start
G0 Z-1                  # Lower pen
G1 F1000 X100 Y10       # Go to end
G0 Z10                  # Raise pen
"""
```

```
plotLine :: Line -> [String]
```

More complex shapes

```
cutPolygon :: Line -> Polygon -> [Polygon]
```




```
polygonEdges :: Polygon -> [Line]
```

```haskell
plotCutPolygon :: Line -> Polygon -> [String]
plotCutPolygon scissors polygon = do
   polygonPart <- cutPolygon scissors polygon
   edge <- polygonEdges polygonPart
   gCodeLine <- plotLine edge
   pure gCodeLine
```

```python
def plotCutPolygon(scissors, polygon):
   for polygonPart in cutPolygon(scissors, polygon):
      for edge in polygonEdges(polygonPart):
         for gCodeLine in plotLine(edge):
            file.append(gCodeLine)
```

Part 3.5: Coding vs. reality

War stories on the subtleties of pen vs. pixel

Mechanical

- Resonances (Flow lines)
- Hysteresis
- Calibration accidents
- Time is a finite resource
- So are pens

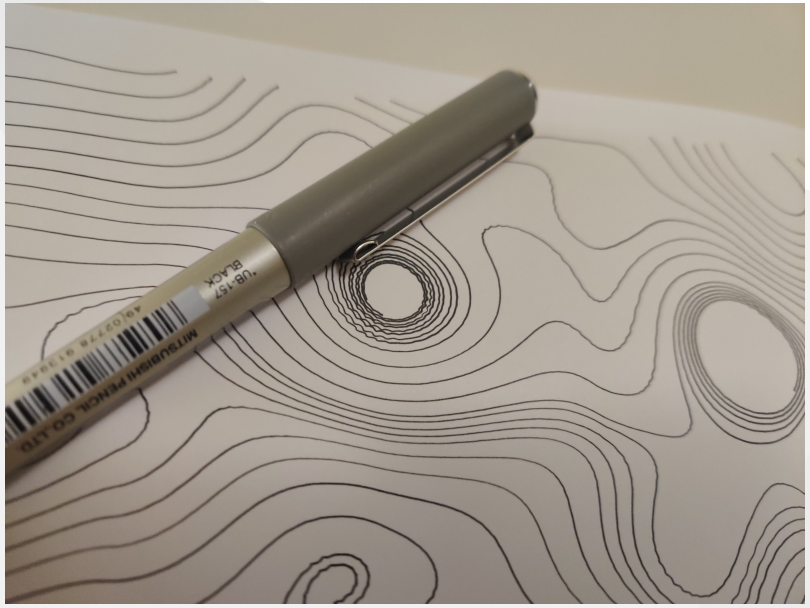
Pens are difficult

- Mechanical abrasion (Vienna)
- Wet paper (Amoeba)
- Ink blots
- Directional pens
- Paper isn’t flat

Part 4: Case study: flow lines

Idea: fake turbulent fluid trajectories

Solution: rediscover fluid dynamics



In a velocity field, where does a little blob of stryofoam move?

```
flowLine :: Vec2 -> (Vec2 -> Vec2) -> [(Double, Vec2)]
flowLine particleStart flowField = solveOde flowField particleStart timeStep maxTime
  where
    timeStep = …
    maxTime = …
```

A flow field that curls more and more on the right

```
velocityField :: Vec2 -> Vec2
velocityField x = Vec2 1 0 + linearRamp 0 picWidth (curl vectorPotential x)

vectorPotential :: Vec2 -> Double
vectorPotential = perlinNoise params
  where
    params = Perlin { frequency = …, seed = …, persistence = …, … }
```

```haskell
type Trajectory = [Line]

picture :: [Trajectory]
picture = for_ [1..picHeight] $ \particleStart -> flowLine particleStart velocityField

plotLine :: Line -> [String]
plotLine (Line (Vec2 x0 y0) (Vec2 y1 y1)) =
    [ "G1 F1000 X{x0} Y{y0}" -- Pretend Haskell has good string formatting
    , "G1 F1000 X{x1} Y{y1}"
    ]

plotTrajectory :: Trajectory -> [String]
plotTrajectory trajectory = penDown ++ plot ++ penUp
  where
    penDown = ["G1 F1000 Z-1"]
    penUp = ["G1 F1000 Z5"]
    plot = map plotLine trajectory

main = do
    render "trajectories-preview.svg" (toSvg picture)
    toGCode "trajectories.g" (unlines (map (unlines . plotTrajectory) picture))
```