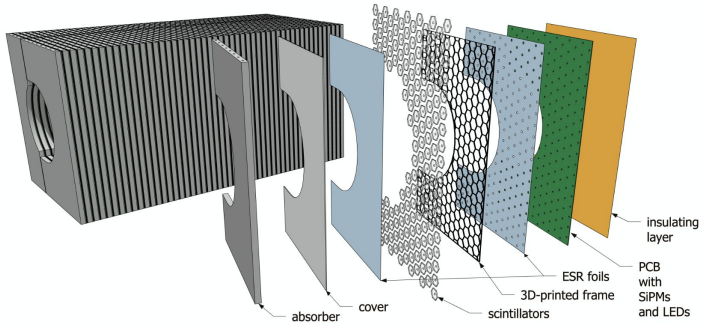
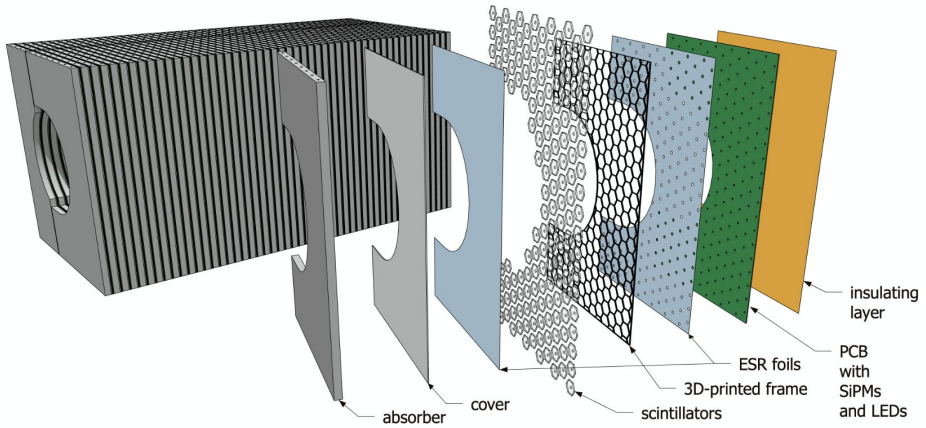
Generating Gerbers Files for the Insert's PCB boards

Sebouh Paul
UC Riverside
8/7/24



Why Algorithmically Generate PCBs?

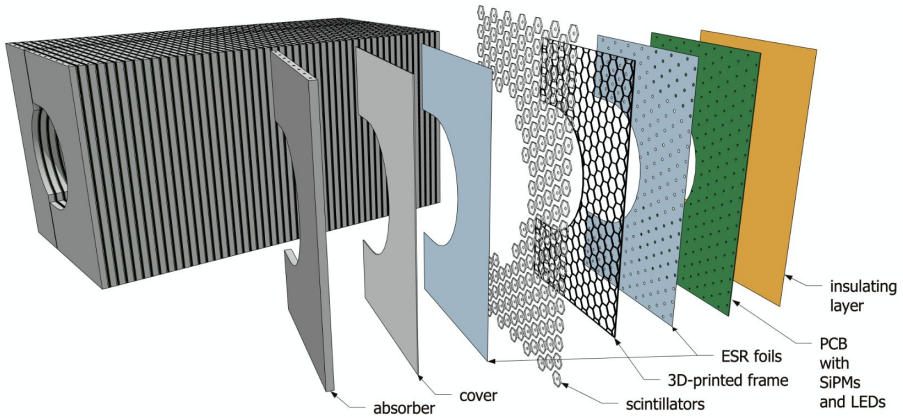
- No two boards are the same:
  - Each PCB layer has a different size hole to accommodate the hole for the beampipe
  - Using “staggering” to improve position resolution (or conversely use larger tiles without sacrificing resolution*) requires multiple board layouts
- Designing 120 boards by hand would be very time consuming



*https://doi.org/10.1016/j.nima.2023.169044

Goals

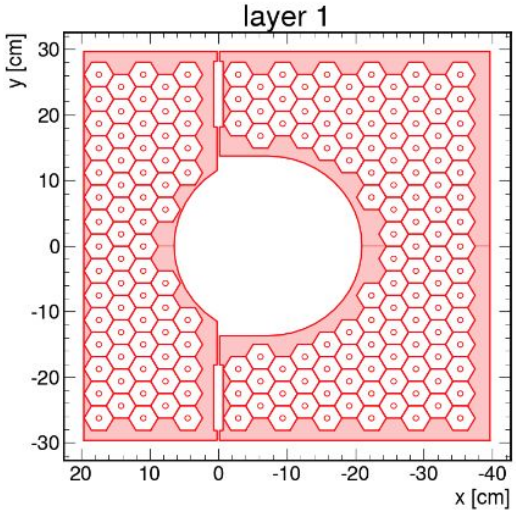
- Staggered layout for high effective granularity*
- Minimize dead space
- Use only complete hexagon cells
- Target of ~6000 total channels

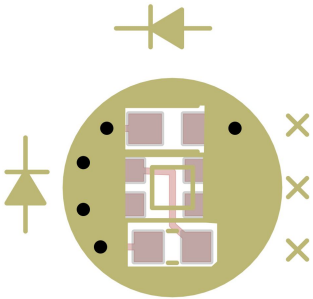


*https://doi.org/10.1016/j.nima.2023.169044

Scripts for Board Generation

- First script:
  - Encodes where the edges of the boards are based on our engineering diagram
  - Determines what the positions of the cell centers are, such that the cells are packed hexagonally, whole cells only
- Second script:
  - Creates edge-cuts based on the edges of the PCB boards
  - Places SiPMs and other components at center of cells.
  - Draws hexagonal cells on the board silkscreen for reference
  - TODO: connectors and traces connecting SiPMs to connectors



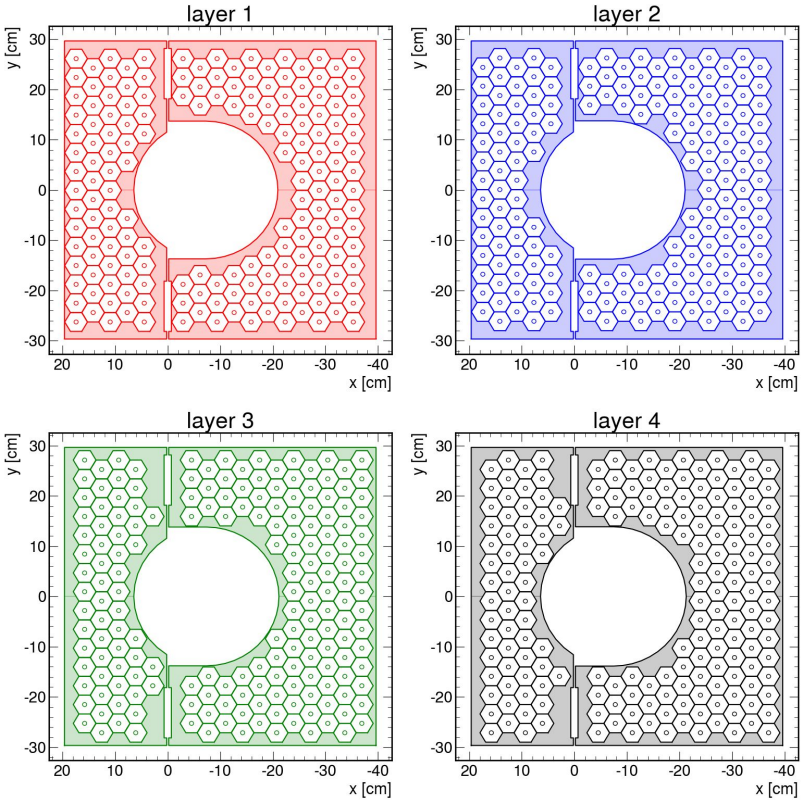


*https://doi.org/10.1016/j.nima.2023.169044

How can we implement this?

First 20 layers:

- High granularity: ~12 cm^2
- Staggered according to H4 pattern
- Goal: determining the position of the start of the showers

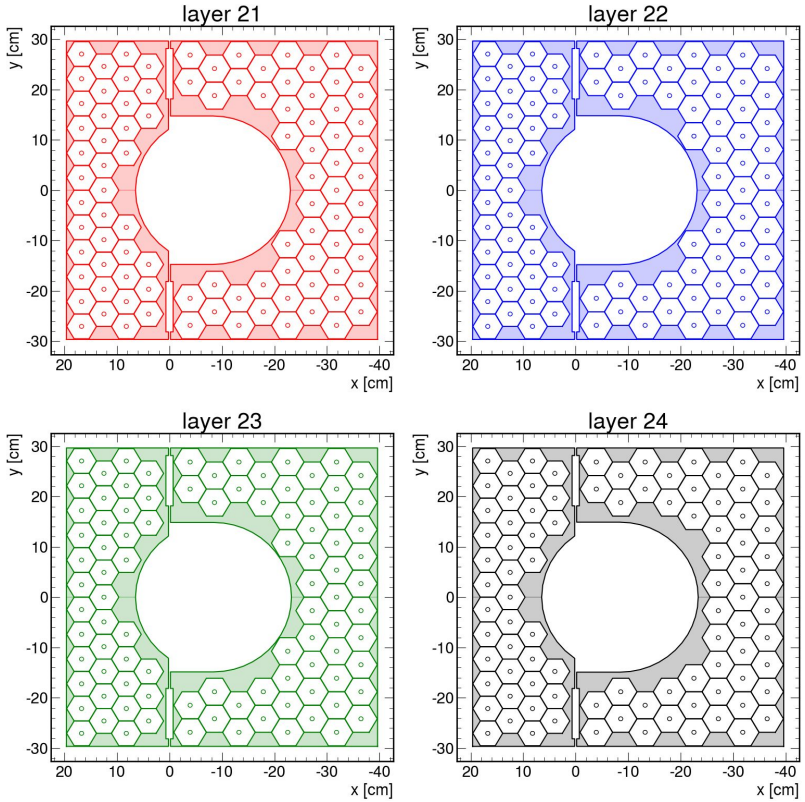


https://github.com/sebouh137/staggered_tesselations/blob/main/GenerateLayersH4.ipynb

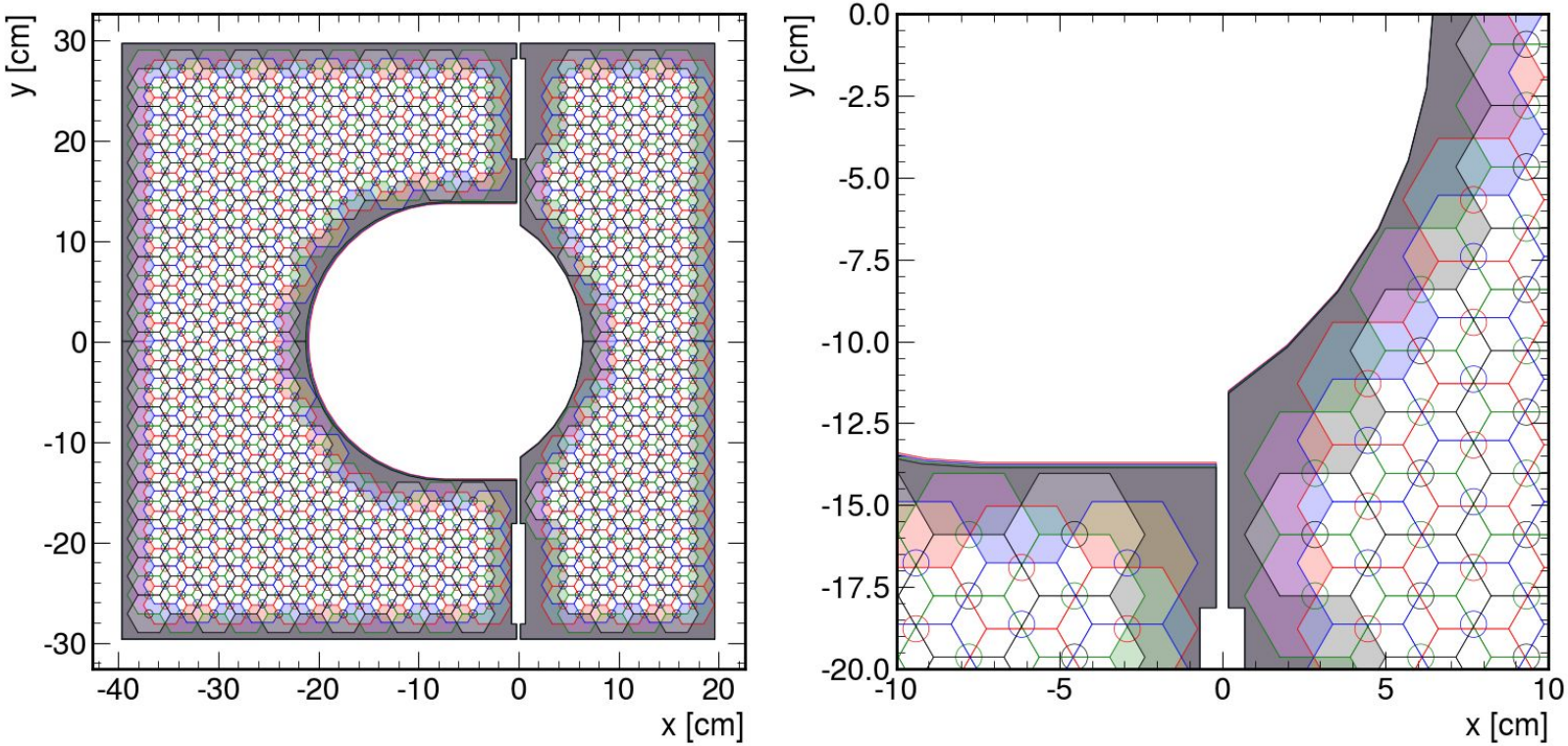
How can we implement this?

Remaining layers (21-60)

- No staggering
- Maximizes the active area

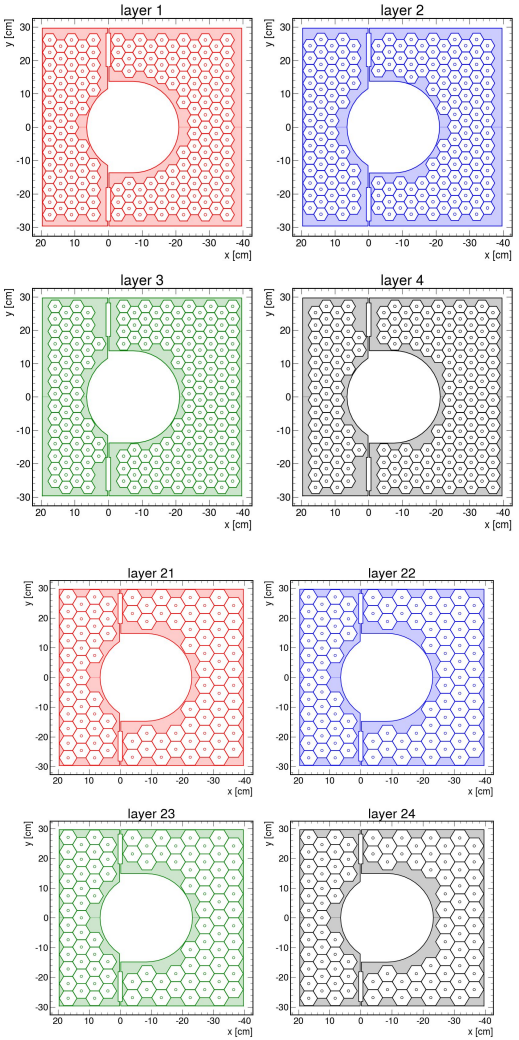


Deadspace in one layer is covered (partially) by cells in other layers through staggering



Statistics of this implementation:

- 20 staggered, high-granularity layers
- 40 unstaggered, low-granularity layers
- Total channels: 7077



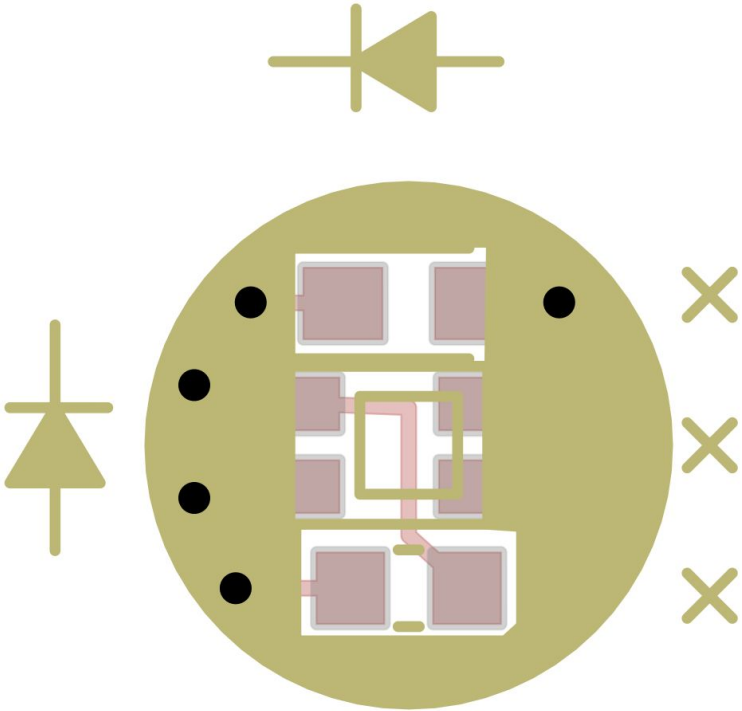
Cell positions and PCB edges can be output to csv files

```
cat insert_layout/SiPM_positions_60L.csv
,x,y,row,col,area
0,8.225,-27.055,1,1,20.955
1,16.745,-27.055,1,3,20.955
2,3.965,-24.595,2,0,20.955
3,12.485,-24.595,2,2,20.955
4,8.225,-22.136,3,1,20.955
5,16.745,-22.136,3,3,20.955
6,3.965,-19.676,4,0,20.955
7,12.485,-19.676,4,2,20.955
8,8.225,-17.217,5,1,20.955
9,16.745,-17.217,5,3,20.955
10,3.965,-14.757,6,0,20.955
11,12.485,-14.757,6,2,20.955
12,8.225,-12.298,7,1,20.955
13,16.745,-12.298,7,3,20.955
14,12.485,-9.838,8,2,20.955
15,8.225,-7.379,9,1,20.955
16,16.745,-7.379,9,3,20.955
17,12.485,-4.919,10,2,20.955
18,16.745,-2.460,11,3,20.955
19,12.485,0.000,12,2,20.955
20,16.745,2.460,13,3,20.955
21,12.485,4.919,14,2,20.955
22,8.225,7.379,15,1,20.955
23,16.745,7.379,15,3,20.955
24,12.485,9.838,16,2,20.955
25,8.225,12.298,17,1,20.955
26,16.745,12.298,17,3,20.955
27,3.965,14.757,18,0,20.955
28,12.485,14.757,18,2,20.955
29,8.225,17.217,19,1,20.955
30,16.745,17.217,19,3,20.955
31,3.965,19.676,20,0,20.955
32,12.485,19.676,20,2,20.955
33,8.225,22.136,21,1,20.955
34,16.745,22.136,21,3,20.955
35,3.965,24.595,22,0,20.955
36,12.485,24.595,22,2,20.955
37,8.225,27.055,23,1,20.955
38,16.745,27.055,23,3,20.955
```

```
cat insert_layout/PCB_edges_60L.csv
,x,y
0,19.630,-29.645
1,19.630,29.645
2,0.190,29.645
3,0.190,28.140
4,0.690,28.140
5,0.690,18.140
6,0.190,18.140
7,0.190,13.124
8,2.030,11.418
9,3.596,9.457
10,4.851,7.284
11,5.768,4.949
12,6.327,2.502
13,6.515,0.000
14,6.327,-2.502
15,5.768,-4.949
16,4.851,-7.284
17,3.596,-9.457
18,2.030,-11.418
19,0.190,-13.124
20,0.190,-18.140
21,0.690,-18.140
22,0.690,-28.140
23,0.190,-28.140
24,0.190,-29.645
25,19.630,-29.645
```

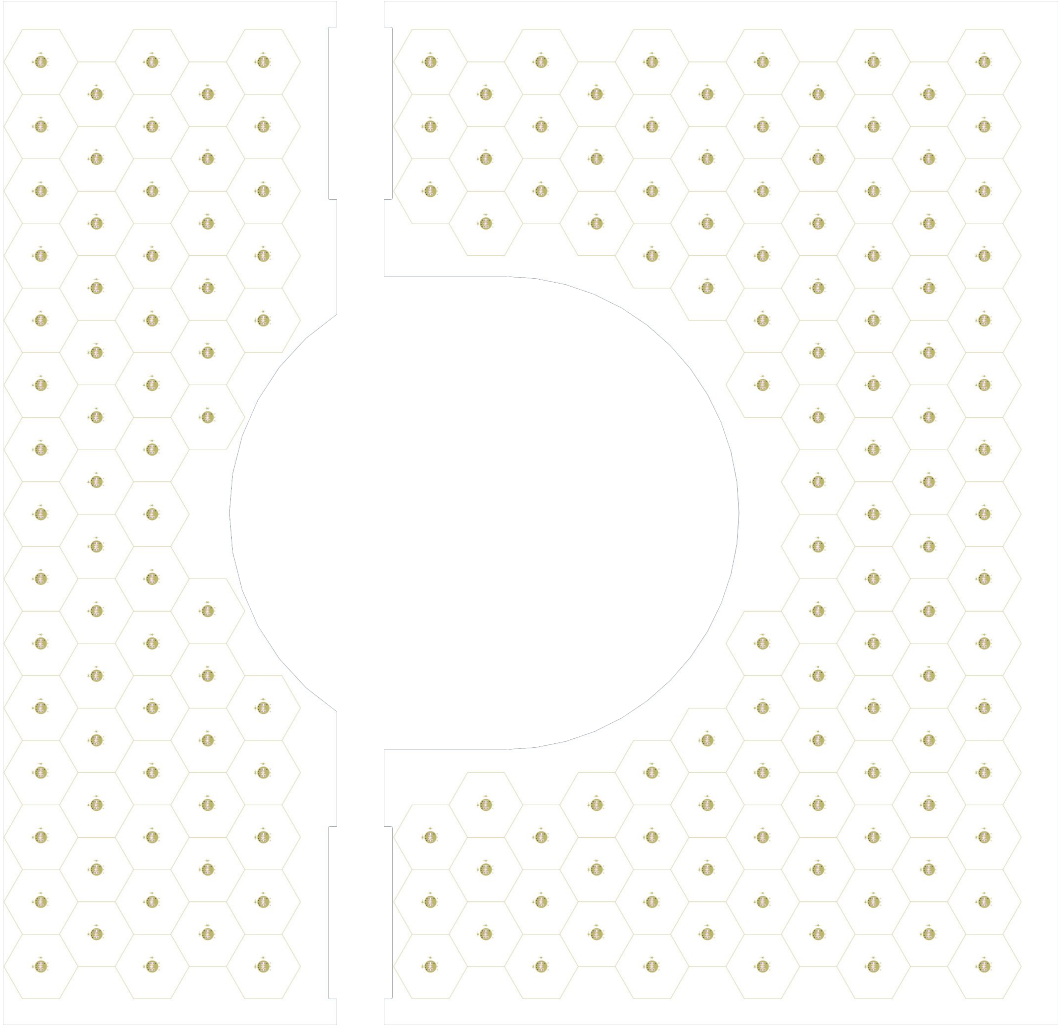
Generating Gerbers files

- pcbflow library in python allows us to create Gerbers files
- Example on right: zoom in on the components in the center of a cell
  - SiPM (center)
  - LED (top)
  - Capacitor (bottom)
- No traces on front active area except inside dimple (circle) so that ESR foil can rest flat on the PCB
  - Pads are connected to vias, allowing traces on back or internal layers to connect SiPM and other components to the



Generating PCBs

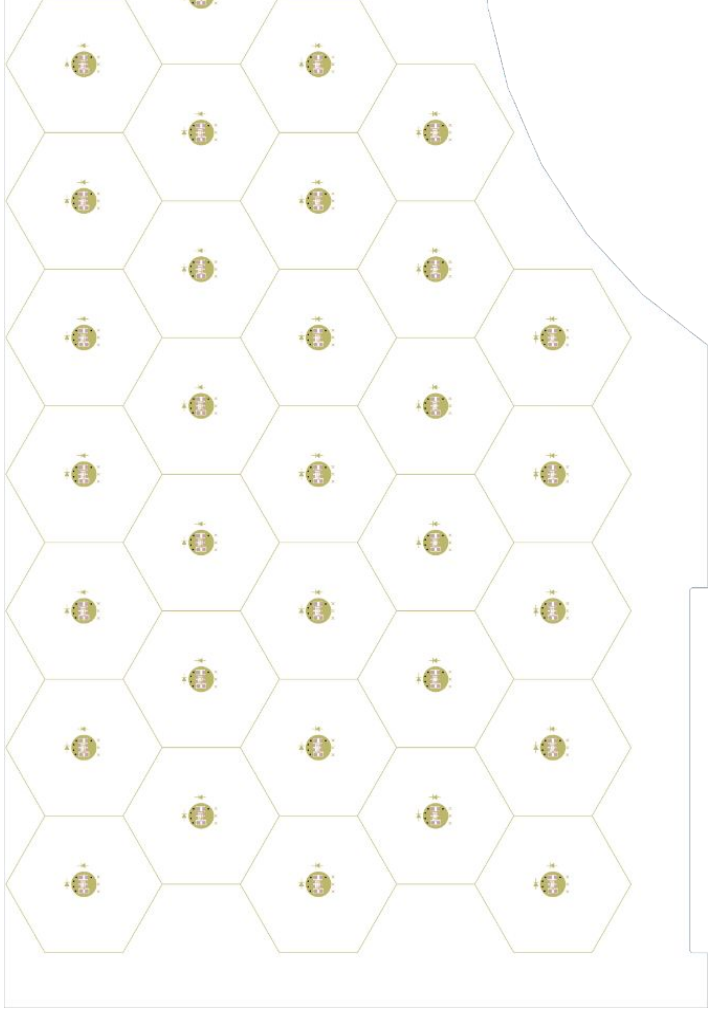
- Python script* allows us to take the positions of cell centers and board edges and convert it to a Gerbers file
- TODO update script to include the following components:
  - Connectors and traces to SiPMs
  - Documentation on silkscreen:
    - Layer numbers, channel numbers, etc.



*https://github.com/sebouh137/staggered_tesselations/blob/main/GeneratePCB.ipynb

Connectors

- When we find out which types of connectors will be used, we will update our Gerbers-making script:
  - Add connectors
  - Add code that connects the connectors to the appropriate SiPMs and other components



Summary

- The layout for the SiPMs on the PCBs can be determined algorithmically
- Current design:
  - Total number of channels is close to the goal of ~6000
  - Only full hexagonal cells
  - High granularity/staggered for first 20 layers
  - Low granularity/unstaggered “tail catcher” for last 40 layers.
- Cell positions and the edges of the PCB are written out to CSV files
- Python code creates Gerbers file for the PCB with SiPMs, and capacitors in each.
- Automatic workflow, which makes complex geometry not ultra time consuming. Preliminary layout subject to minor tweaks as needed.