MECE E4606
[Digital Manufacturing]

Assignment 5 – Food Printing

Group 7: Rayal Raj Prasad (rpn2108) Jacob Joseph (jcj2131) Jia Huang (jmh2299) Brian Jin (bj2364)

Table of Contents

Part I: 1-inch Nutella Square

Section I: Description

Originally the G-Code was written in inches and had to be converted to millimeters because troubleshooting on the printer was easier. 1 inch is equal to 25.4 mm which can be seen in the G-code in Section II. The suggested feed rate was 400 mm/minute and is set as such. The extrusion rate was calculated from the ratio of the syringe tube to the syringe tip (16 gauge) and multiplied by the distance traveled per command (25.4 mm). A priming distance of about 28mm was used to ensure the material was flowing perfectly before starting the square.

There were a couple of problems with the initial print. The Z-axis was set too high and the sides of the square came out stretched and uneven. In the second trial, the Z-axis was too low and the material was spread and the sides came out too flat. The third try proved successful as an appropriate distance of the extruder tip from the printing surface was chosen. The sides came out perfectly with uniform thickness.

Section II: G-Code

The G-Code used to draw the 1-inch square of Nutella was as follows:

```
G91

G21

G92 X0 Y0 Z0 E0

G01 X20 Y20 Z0 E0.35 F400 ; Initial priming line

G01 X25.4 Y0 Z0 E0.35

G01 X0 Y25.4 Z0 E0.35

G01 X-25.4 Y0 Z0 E0.35

G01 X0 Y-25.4 Z0 E0.35

G01 X-20 Y-20 Z0 E0.35 ; Final priming line

M84
```

Note that the extruder is first primed by drawing a line before the actual square is drawn. Once the square is completed, the extruder is brought back through the same priming line to prevent Nutella accumulating at the corner of the square.

Section III: Path Simulated

There are several free G-Code simulators available on the internet. A simple program to simulate G-Code was found on GitHub[1] and was using to simulate the code. Fig. 1 and 2 show some screenshots of the simulation. Note the priming line that is traced before and after the square is printed.

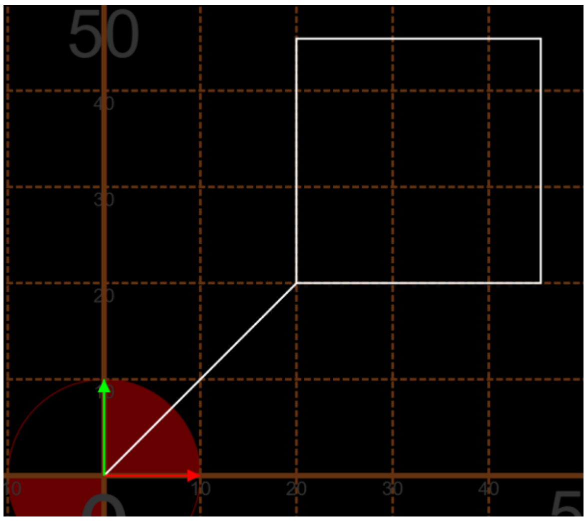

Figure 1: G-Code simulation in X-Y plane

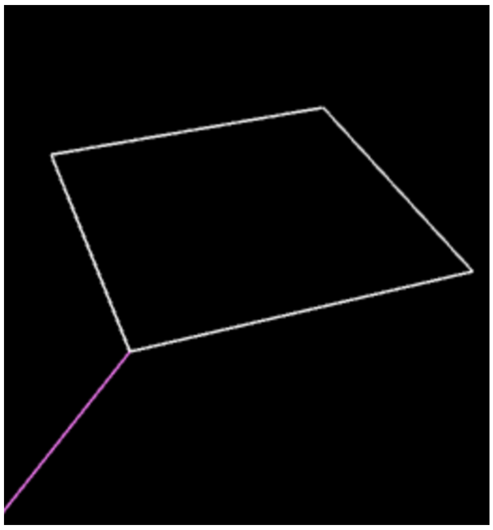

Figure 2: G-Code simulation in 3-D

Section IV: Close-up Shots and Video

The square was drawn using the G-Code in Section II. Fig. 3 shows the print in progress while Fig. 4 shows the finished print using Nutella. A video of the entire process was also recorded[2]. It was also uploaded to YouTube (Link: https://youtu.be/yhwfq5PD0UQ).

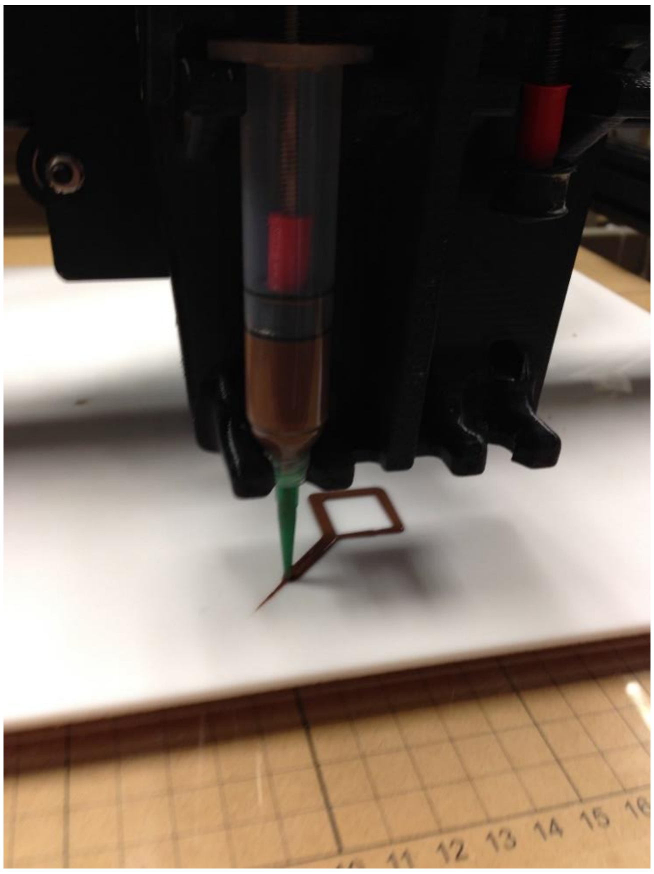

Figure 3: Print in progress

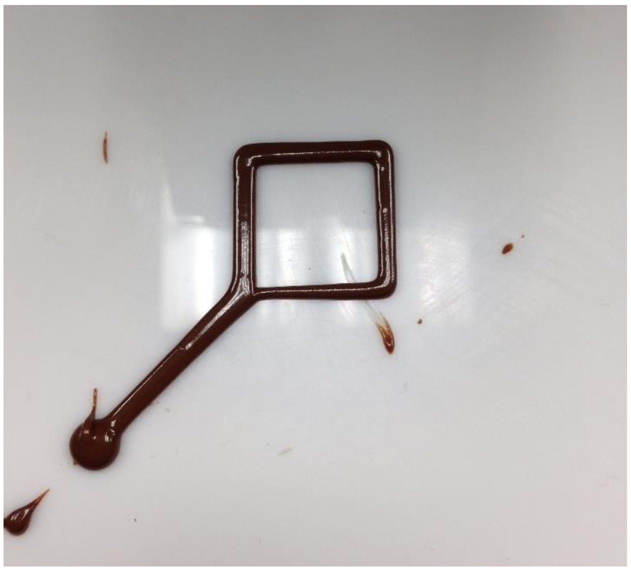

Figure 4: Finished print

Part II: Twisted Triangle 3-D Structure

Section I: Description

Role Distribution

A simplified RACI matrix was formulated to easily explain the distribution of the roles within the team.

	Brainstorming	Recipe Design & Purchase	Coding - Structure Design	Coding for G-Code Generation	Printer and material setup	Printing and Troubleshooting
Brian Jin	R	P	R	P	R	R
Jacob Joseph	R	P	P	R	R	R
Rayal Raj Prasad	R	P	R	P	R	R
Jia Huang	R	R	P	P	R	R

Legend

P	Participating
R	Responsible
X	Neither participating nor responsible

Process Approach

A MATLAB program was written to generate the G-Code of a three-dimensional twisted triangle structure with the following input parameters:

- Height
- Length of sides
- Tolerance for spacing between adjacent walls
- Extrusion nozzle size
- Syringe Diameter
- Number of walls
- Degree of rotation per layer

- Scale factor to reduce side length of each layer as height increases
- Center of the structure

The structure was first designed and plotted in MATLAB to make sure that the coordinates were accurate.

Algorithm

The algorithm to generate the tool path and G-Code is as follows.

To initialize the triangle, it was assumed that the bottom side would be the longest, and the triangle sides would be sorted counter-clockwise in descending lengths. The bottom left point was initially considered as the origin, and using the law of sines, the angles were determined. Once this was done, the centroid of this triangle was computed, and comparing it with the desired centroid (center), the triangle was mapped to the desired layer.

The start point of the first layer was found taking the base layer dimensions of the triangle and subtracting the required wall thickness. The inner wall bottom left vertex was labeled as the first vertex of the current layer. Side one is drawn to the second vertex and side two was drawn to the third vertex in a counter-clockwise fashion. The third side in a triangle would be drawn from the third vertex back to the first. With multiple walls the innermost triangle was drawn first, then the third vertex of the inner triangle was connected to the first vertex of the outer adjacent triangle, and so on. Once the outermost triangle was complete, a new layer was started by connecting the third vertex of the outer triangle with the first vertex of the innermost triangle of the next layer up. This allowed for the toolpath of the printer to be continuous from point to point throughout the entire print and the process was repeated until the specified height was reached.

Additional features built into the program include a priming line to start the extrusion process before the main structure is printed as well as an ending line taking away the extrusion head from the print while the extrusion has stopped. Both account for the delay that occurs when the material starts/stops extruding. Absolute coordinates were used and the zero location was made to be where the print head was located at the start of the printing process.

The program was made to generate the G-Code file directly based on the above input parameters chosen by the user. A four dimensional vector for each vertex of every triangle in the structure was written into one line of G-Code that specifies the x, y, and z coordinates as well as the extrusion amount. A loop is run to generate every line of G-Code that will generate the tool path for the printer.

Problems Encountered

One of the biggest problems faced was filling the syringe with material without air pockets forming. Because the material used was fairly viscous it was difficult to compact all of it at the head of the syringe. The first method used was loading with a popsicle stick and working out any air pockets that may have formed. Another method used was filling a cone shaped cake icing bag with the material, cutting a hole in the tip and placing it at the bottom of the syringe. Then the icing was extruded out of the bag into the syringe filling it from the bottom up. This technique was fairly successful but air pockets were still encountered when printing.

When first printing the center of the structure was placed at the absolute zero location of the printing system. The generated g-code had negative x and y coordinates which were not being read by the machine. It was found that the machines absolute coordinates do not allow negative values and so the center location the piece had to be moved to allow for all positive coordinates.

Another situation arose when the print head was set over the center of the machine workspace. Because the start location of the head is set to be the zero location of the machine absolute coordinates the offset to the start of the structure was too far and the parts of the machine hit the shelf lying inside its workspace. Going forward each time before starting the program the extrusion head was placed in the left-most corner of the workspace.

During the first iteration only one triangle and one side length was used for each layer. After 10 layers the walls started to sag and eventually collapsed. The material was not strong enough to support the upper layers and so more walls were added. Using two and three walls the structure was able to support 15 layers but sagging still occurred near the center of each side.

Lastly the size of each triangle was decreased for each successive layer going upwards. This inward wall angle provided more support and decreased the amount of sagging present in the center of each wall. However, this center wall location for each triangle was still lower than the corners. Going forward the extrusion rate can be modified to extrude less at the corners and more near the center of the sides.

Section II: MATLAB Source Code (with comments)

```matlab
% MATLAB code to generate G-code to be input to the food printer to print a
% twisted triangle structure. Note that to run this code, MATLAB R2016b or
% later is required to support local function calling.
%
% Course: Digital Manufacturing - Dr. Hod Lipson
% Creators: Jacob Joseph, Rayal Raj Prasad, Brian Jin, Jia Huang
% Department of Mechanical Engineering, Columbia University in the City of New York

close all
clc

gauge = 1.29;    % 16-gauge nozzle
syringe = 16;    % syringe diameter
tol = .05;       % tolerance to place adjacent walls
scale_factor = 0.975;   % scaling of sides as structure increases in height
t = gauge - tol;    % increment of each height accounting for fudge

x0 = 60;         % centroid of structure
y0 = 60;
z0 = 0;
center = [x0;y0;z0];

l1 = 25;         % setting length of sides
l2 = 25;
l3 = 25;
L = sort([l1 l2 l3]);   % sorting triangle lengths
h = 25; % setting height of the final structure
theta = pi/150; % angle rotation step per layer of the structure

% calculating the angles of the triangle using law of sines
phi1 = acos((-L(3)^2 + L(2)^2 + L(1)^2)/(2*L(2)*L(1)));
phi2 = asin(sin(phi1)*L(2)/L(3));
phi3 = asin(sin(phi1)*L(1)/L(3));

n = 3;  % number of walls

% computing perpendicular distance between adjacent walls
ext_D = gauge+tol*2+1.5;
ds1 = ext_D/(sin(phi1));
ds2 = ext_D/(sin(phi2));
ds3 = ext_D/(sin(phi3));

% decrementing length of sides so that the structure starts at inner wall
L = L - n*[ds1 ds2 ds3];

z = linspace(0,h,h/t);  % defining layers

P = zeros(n*3*size(z,2),3); % creating empty matrix to store points

k = 1;  % counter variable

% loop to initialize points of all walls of the base layer
for i = 1:n

    L = L + [ds1 ds2 ds3];  % traversing outward from inner wall
    P1_temp = [0,0,0];  % bottom left vertex of triangle set as origin
    P2_temp = [L(3),0,0];   % bottom side of triangle parallel to x-axes
    P3_temp = [L(1)*cos(phi2),L(1)*sin(phi2),0]; % computing 3rd point

    % computing the centroid using the midpoint of the base and the
    % third vertex
    basemid = [0.5*(P1_temp(1)+P2_temp(1)),0.5*(P1_temp(2)+P2_temp(2))];
    centroid = [P3_temp(1) + (2/3)*(basemid(1)-P3_temp(1)),P3_temp(2) ...
```

```
        + (2/3)*(basemid(2)-P3_temp(2)),0];

    % offset between computer centroid and desired value
    offset = center' - centroid;

    % mapping triangle to desired centroid
    P1 = (offset+P1_temp);
    P2 = (offset+P2_temp);
    P3 = (offset+P3_temp);

    % storing points in the P matrix
    P(k:k+2,:) = [P1;P2;P3];

    k = k + 3; % incrementing by 3 as 3 points were computed
end

% loop to rotate the triangle as the height increases
for i = 1:size(z,2)

    for j = 1:n

        % calling a rotate function to rotate by pi/150
        P1 = RotZTheta(P(k-n*3,:)'+[0;0;t],theta,center);
        P2 = RotZTheta(P(k-n*3+1,:)'+[0;0;t],theta,center);
        P3 = RotZTheta(P(k-n*3+2,:)'+[0;0;t],theta,center);

        P(k:k+2,:) = [P1';P2';P3'];

        k = k + 3;

    end
end

P(1,4) = 0; % creating a new column for length of side from previous point

for i = 2:size(P,1)
    % computing length between current and previous point (0 for 1st point)
    P(i,4) = pdist([P(i,1:3);P(i-1,1:3)],'Euclidean');
end

P_xy = P(:,1:2);    % temporarily extract x and y values for scaling
P_xy = P_xy - repmat([x0 y0],size(P,1),1);  % mapping back to origin
diff = zeros(size(P_xy));   % matrix to store difference caused by scaling

% scaling of the outer wall triangle in each layer
for i=1:size(P,1)/(3*n)

    % computing start and end indices for each layer
    r_start = 1+(3*n*(i-1));
    r_end = r_start + 3*n - 1;

    % exponentially scaling only the outer triangular wall of the layer
    P_xy(r_end-2:r_end,:) = P_xy(r_end-2:r_end,:) * scale_factor^(i-1);

    % storing the difference to help scale the inner walls
    diff(r_end-2:r_end,:) = P(r_end-2:r_end,1:2) - P_xy(r_end-2:r_end,:)...
        - repmat([x0 y0],3,1);


end

% scaling of all walls to ensure no intersection
for i=1:size(P,1)/(3*n)

    % start and end indices for each layer
    r_start = 1+(3*n*(i-1));
    r_end = r_start + 3*n - 1;
    % scaling back the inner walls by the difference to ensure no collision
    P_xy(r_start:r_end-3,:) = P_xy(r_start:r_end-3,:)...
        - repmat(diff(r_end-2:r_end,:),n-1,1);

end

% mapping points back to desired centroid
P(:,1:2) = P_xy + repmat([x0 y0],size(P,1),1);

% plotting required G-Code path
figure
plot3(P(:,1),P(:,2),P(:,3))

% call function to write G-Code file
writetogcode(P,gauge,syringe)

function writetogcode(V,gauge,syringe) % function to write G-Code file

    fid = fopen('triangle.gcode','w'); % open a new file "triangle.gcode"
    fprintf(fid,'M106 S200 \n');     % turn on fan
    fprintf(fid,'G90 \n');  % absolute coordinate system
    fprintf(fid,'G21 \n');  % set to millimeters
    fprintf(fid,'G92 X0 Y0 Z0 E0 \n');  % set current position to zero
    fprintf(fid,'G0 X0 Y0 Z0 E0 \n');   % move head to zero position

    L = V(:,4); % extracting lengths from point matrix
    l = 50; % for priming line
    ratio = (gauge^2)/syringe^2; % computing extrusion ratio
    E1 = ratio*l;   % extrusion needed for priming line
    E = extrusion(V,gauge,syringe,L); % function to give a vector for E

    % priming path start and end points
    Xstart = V(1,1)-l;
    Ystart = V(1,2);
    Zstart = V(1,3);
    fprintf(fid,'G01 X%f Y%f Z%f E0 F400 \n',Xstart,Ystart,Zstart);

    Xend = V(1,1);
    Yend = V(1,2);
    Zend = V(1,3);
    fprintf(fid,'G01 X%f Y%f Z%f E%f \n',Xend,Yend,Zend,E1);

    % running through all the points in the matrix and writing to file
        for i = 2:size(V,1)
            X = V(i,1);
            Y = V(i,2);
            Z = V(i,3);
            fprintf(fid,'G01 X%f Y%f Z%f E%f \n',X,Y,Z,E(i)+E1);
        end

    % ensuring that the extrusion head stays at the same Z but moves away
    fprintf(fid,'G01 X%f Y%f Z%f E0 \n',Xstart,Ystart,V(end,3));
    fprintf(fid,'M84 \n'); % shuts off motors
    fclose(fid);
end

function [P] = RotZTheta(point,theta,center)    % function to rotate points

    P0 = point - center;    % vector from point to center

    % rotation matrix for rotation about z-axis
    R = [cos(theta) -sin(theta) 0; sin(theta) cos(theta) 0; 0 0 1];

    % rotation + translation
    P = R*P0+center;
```

```
end

function E = extrusion(V,gauge,syringe,Length) % function to compute E

    ratio = (gauge^2)/syringe^2; % extrusion ratio

    E = zeros(1,size(V,1)); % empty matrix to store E values

    % running through matrix to compute E using length and ratio
    for i = 2:size(V,1)
        e = ratio.*Length(i);
        E(i) = e+E(i-1);  % incrementing to use with absolute coordinates
    end
end
```
Published with MATLAB®

When this program is run with the right parameters assigned to the variables at the beginning of the code, the program generates two outputs:

1. G-Code file in the current path titled “triangle.gcode”.
2. A plot of the G-Code path (shown in Fig. 5 for the input parameters in Section III)

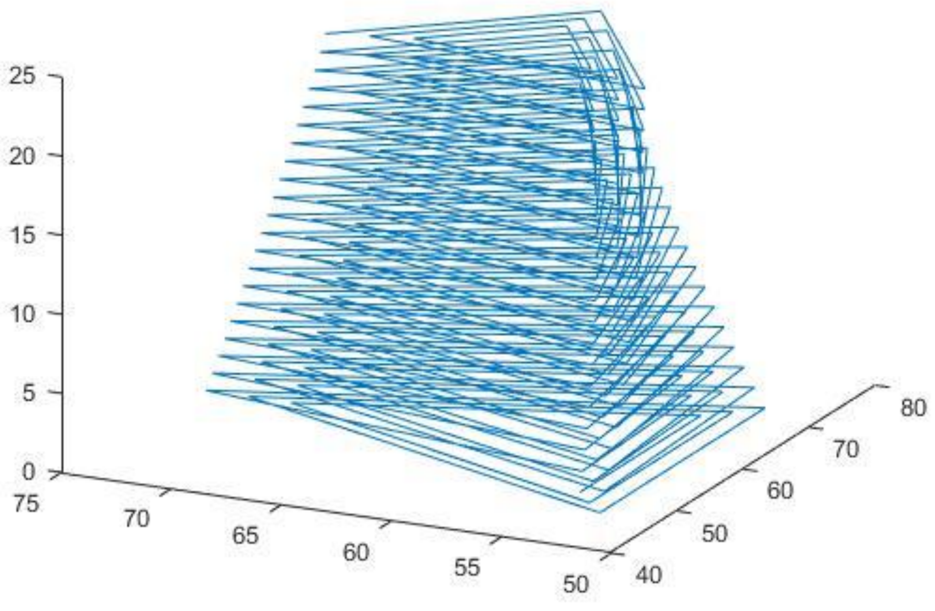

Figure 5: Plot generated from MATLAB code

Section III: G-Code

The MATLAB code was used to generate the G-Code for the twisted triangle structure with the following parameters:

- Height: **25 mm**
- Length of sides: **25 mm** equilateral triangle
- Tolerance: **0.05 mm**
- Extrusion nozzle size: **16 gauge** (1.29 mm)
- Syringe Diameter: **16 mm**
- Number of walls: **3**
- Degree of rotation per layer: **π/160**
- Scale factor for reducing side length: **97.5%**
- Center of the structure: **(60,60,0)**

The G-Code generated is shown below:

```
M106 S200
G90
G21
G92 X0 Y0 Z0 E0
G0 X0 Y0 Z0 E0
G01 X0.837085 Y54.709788 Z0.000000 E0 F400
G01 X50.837085 Y54.709788 Z0.000000 E0.325020
G01 X69.162915 Y54.709788 Z0.000000 E0.444145
G01 X60.000000 Y70.580423 Z0.000000 E0.563270
G01 X49.168542 Y53.746455 Z0.000000 E0.693392
G01 X70.831458 Y53.746455 Z0.000000 E0.834209
G01 X60.000000 Y72.507090 Z0.000000 E0.975026
G01 X47.500000 Y52.783122 Z0.000000 E1.126819
G01 X72.500000 Y52.783122 Z0.000000 E1.289329
G01 X60.000000 Y74.433757 Z0.000000 E1.451839
G01 X51.258537 Y54.705982 Z1.240000 E1.594261
G01 X68.955486 Y55.076680 Z1.240000 E1.713386
G01 X59.785977 Y70.217338 Z1.240000 E1.832511
G01 X49.610535 Y53.707916 Z1.240000 E1.962633
G01 X70.643837 Y54.148501 Z1.240000 E2.103450
G01 X59.745628 Y72.143582 Z1.240000 E2.244268
```

```
G01 X47. 962533 Y52. 709851 Z1. 240000 E2. 396061
G01 X72. 332188 Y53. 220323 Z1. 240000 E2. 558570
G01 X59. 705279 Y74. 069826 Z1. 240000 E2. 721080
G01 X51. 668377 Y54. 712592 Z2. 480000 E2. 863502
G01 X68. 744841 Y55. 428307 Z2. 480000 E2. 982627
G01 X59. 586781 Y69. 859101 Z2. 480000 E3. 101752
G01 X50. 041639 Y53. 680232 Z2. 480000 E3. 231874
G01 X70. 452260 Y54. 535690 Z2. 480000 E3. 372692
G01 X59. 506101 Y71. 784078 Z2. 480000 E3. 513509
G01 X48. 414901 Y52. 647872 Z2. 480000 E3. 665302
G01 X72. 159679 Y53. 643073 Z2. 480000 E3. 827812
G01 X59. 425420 Y73. 709054 Z2. 480000 E3. 990321
G01 X52. 066651 Y54. 729095 Z3. 720000 E4. 132743
G01 X68. 531412 Y55. 764970 Z3. 720000 E4. 251868
G01 X59. 401937 Y69. 505935 Z3. 720000 E4. 370993
G01 X50. 461889 Y53. 662894 Z3. 720000 E4. 501115
G01 X70. 257150 Y54. 908306 Z3. 720000 E4. 641933
G01 X59. 280961 Y71. 428799 Z3. 720000 E4. 782750
G01 X48. 857127 Y52. 596693 Z3. 720000 E4. 934543
G01 X71. 982888 Y54. 051643 Z3. 720000 E5. 097053
G01 X59. 159984 Y73. 351664 Z3. 720000 E5. 259563
G01 X52. 453411 Y54. 754984 Z4. 960000 E5. 401985
G01 X68. 315612 Y56. 086970 Z4. 960000 E5. 521110
G01 X59. 230977 Y69. 158046 Z4. 960000 E5. 640235
G01 X50. 871330 Y53. 655409 Z4. 960000 E5. 770357
G01 X70. 058912 Y55. 266635 Z4. 960000 E5. 911174
G01 X59. 069758 Y71. 077956 Z4. 960000 E6. 051992
G01 X49. 289249 Y52. 555834 Z4. 960000 E6. 203784 ...
```

Section IV: Path Simulated

The G-Code was input to online simulator[3] and the tool path was simulated Fig. 6, 7 and 8 show the path simulated on the simulator. Note the decreasing side lengths as the height increases.

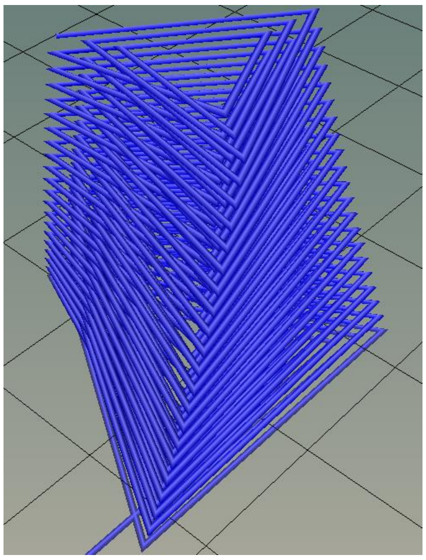

Figure 6: Simulated Path - I

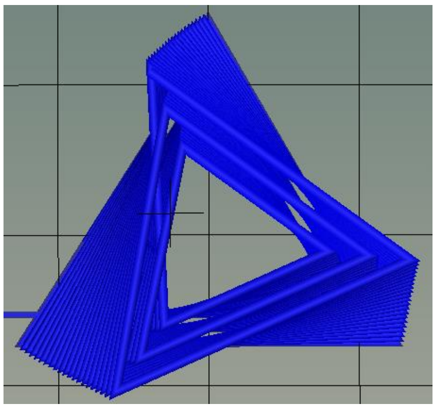

Figure 7: Simulated Path - II

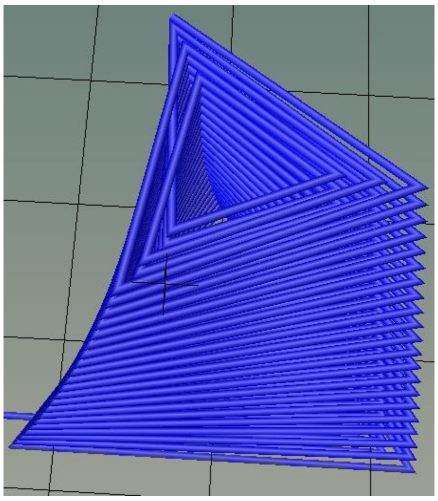

Figure 8: Simulated Path - III

Section V: Close-up Shots

Iteration 1

Fig. 9 shows the first iteration of the print. Each edge of the equilateral triangle was 50 mm in length, and the wall thickness was one layer thick. The print lasted 12 layers before collapsing. The walls began to sag and buckle as the layers were added on. The most sag was seen close to the center of each edge.

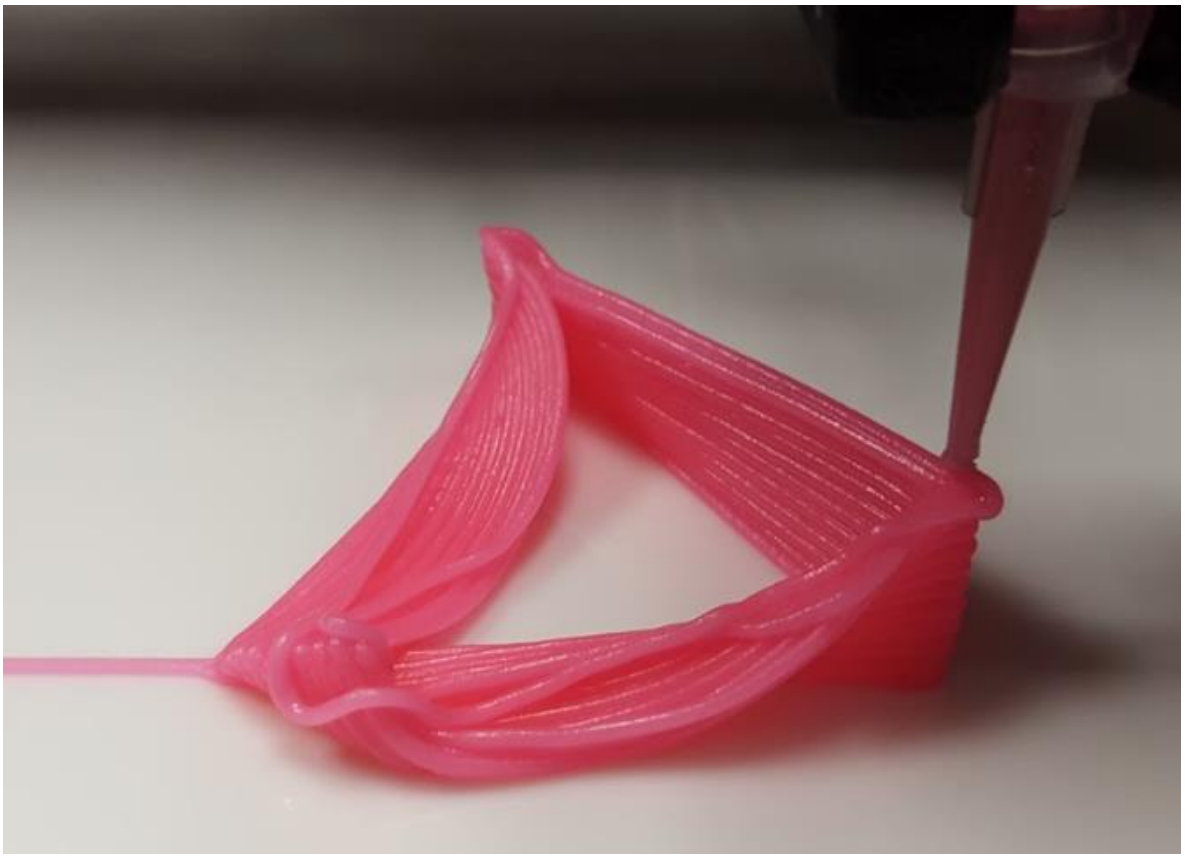

Figure 9: First iteration of print

Iteration 2

For the second iteration, each edge was shortened to 30 mm in length. The wall thickness was also increased to two layers, meaning that two triangles were printed per layer, one inside the other. This meant that a total of 16 layers were printed. The over quality looked decent. However, a few minor farts (repulsive noises caused due to air gaps being encountered while extruding) were experienced. The last layer also had a "curly" look to it, and this could be due to the material being extruded at a height that was above the optimal value, causing the material to "curl" before stacking onto the previous layer. Fig. 10 and 11 show the result of the second iteration of the print.

Iteration 3

For the third iteration, the parameters were changed to include three walls and a decreasing side length as the structure increased in height. The height itself was set to 25 mm, and the sides of the triangle were also set to 25 mm each. The structure held much better than the previous two iterations, and the resultant structure was good. There was still a bit of sagging as the height increased due to the weight of the material. Finally, a huge air bubble was encountered right at the end, causing inconsistencies in the uppermost layer. Fig. 12 and 13 show the end result of the third iteration of the print. Note how the triangle scales in size as the height of the layer increases, and also note the priming line.

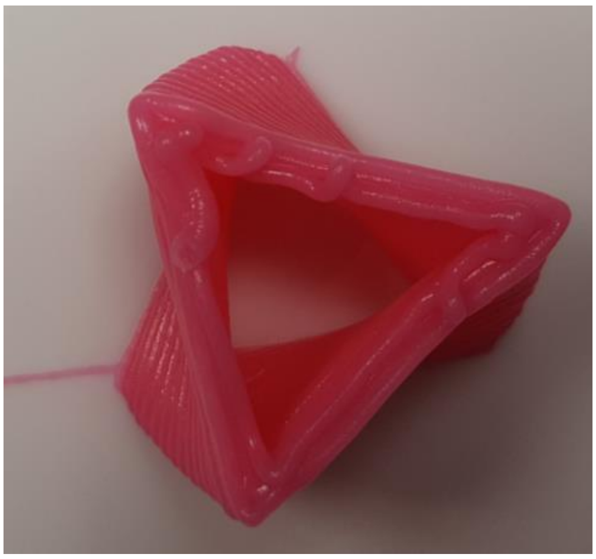
Figure 10: Second iteration of print - I

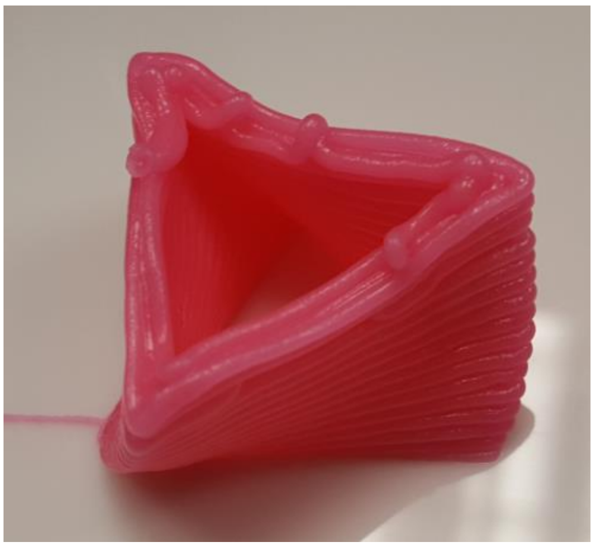
Figure 11: Second iteration of print - II

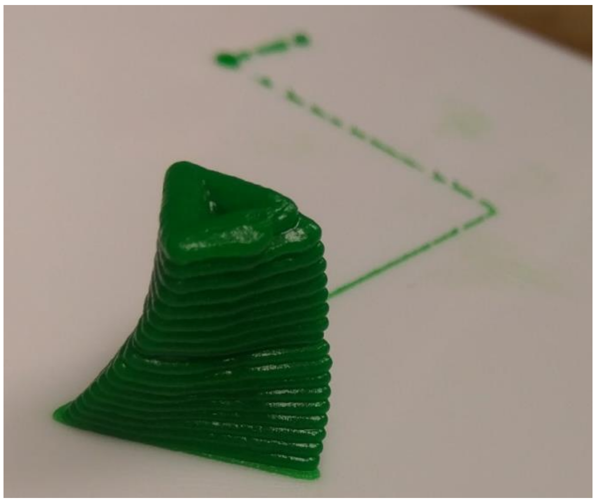
Figure 12: Third iteration of print - I

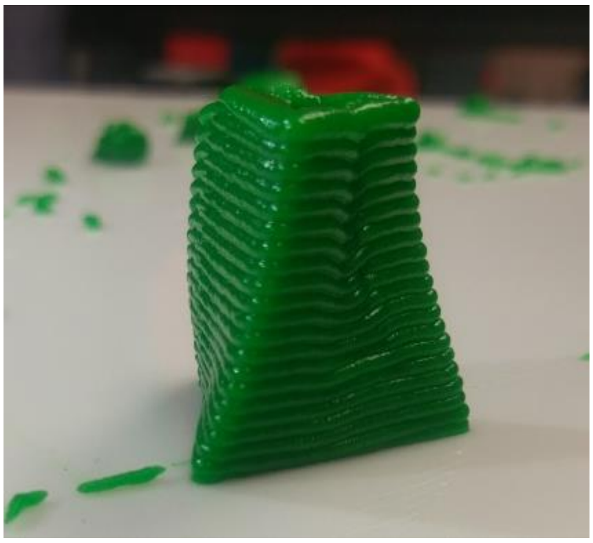
Figure 13: Third iteration of print - II

Section VI: Videos

There were many videos shot of the process through the iterations and uploaded to YouTube.

(recommend watching at a higher speed using video settings on YouTube)

First successful print with single wall, no scaling – Link (https://youtu.be/oeOuhdkjLcs)

Final print – Link (https://youtu.be/0ZWjjlrc4XQ)

Part III: Hypocycloid Shape

Section I: Description

This part includes source code written in MATLAB that generates a G-code file that prints a hypocycloid shape. Input parameters such as size of circle radii, number of loops, number of walls, and number of layers are included to let the user print a customized shape. A number of iterations were done to optimize parameters and materials to allow for the best print possible.

Role Distribution

A simplified RACI matrix was formulated to easily explain the distribution of the roles within the team.

	Brainstorming	**Recipe Design & Purchase**	**Coding - Structure Design**	**Coding for G-Code Generation**	**Printer and material setup**	**Printing and Troubleshooting**
Brian Jin	R	P	R	P	R	R
Jacob Joseph	R	P	P	R	R	R
Rayal Raj Prasad	R	P	R	P	R	R
Jia Huang	R	R	P	P	R	R

Legend

P	Participating
R	Responsible
X	Neither participating nor responsible

Process Approach

The equations describing an epicycloid and hypocycloid[4] were used to generate the tool path for the print. For this assignment, a hypocycloid path is selected because a Spirograph draws the same shape. A hypocycloidal trajectory is drawn using two circles of different radii with the smaller circle tangent to the larger circle and contained inside the larger circle. A point is placed on the smaller circle and the path of the point is traced out as the smaller circle rotates about the larger circle. The equation for this trajectory can be expressed parametrically as,

$$x = r(k-1)cos(kt) + a\ cos((k-1)t)$$

$$y = r(k-1)sin(kt) - a\ sin((k-1)t)$$

where r is the radius of the small circle, k is the ratio of R (larger circle radius) to r, and t the parameter that correlates with the number of loops of the hypocycloid shape.

Section II: Algorithm

The algorithm to generate the toolpath and G-Code is as follows.

First, the parameters to generate the hypocycloid curve must be given. In addition, the user can specify the number of layers and the number of walls for each layer. The walls are generated by creating another hypocycloid curve with each radius scaled by a factor such that the distance between the center of each wall is equal to the gauge size of the tip. With the parameters for the hypocycloid structure given, the code will generate the points of the structure by looping through each wall and each layer. To ensure that the tool path is continuous after each wall is printed, the hypocycloid points for each wall are generated in reverse order after every other wall. This is done using the flip function in MATLAB and the counter variable 'f', which counts the number of walls that have been printed.

In addition to generating the hypocycloid structure, another feature of this code is dual extrusion. This feature was necessary because a single syringe did not have enough volume to extrude a tall hypocycloid structure. The nature of the hypocycloid curve and the gauge of the syringe required a large cross section with multiple walls to produce a visually appealing curve and structure.

The dual extrusion algorithm switches from one syringe to the next syringe when the first syringe runs out of material. The algorithm works by looping through each row of the P matrix, which contains the list of points for the entire hypocycloid structure, and calculating the volume of material extruded after each line. The volume is then compared to the volume of a syringe. When the extruded volume is greater than 80% of the syringe volume, the extruder will finish the layer that it is working on and then switch to the second extruder. Inside the main code, an indicator for the extruder switch (represented by a vector of zeros) is inserted into the P matrix. The "writetogcode" function then searches for the indicator and executes a tool switch when the indicator is found. During the tool switch the first tool head is returned back to the zero location. It then returns to the priming line but before the priming starts the head shifts 51 mm in the negative x direction to offset for the distance between tool 1 and tool 2. Priming of the second tool begins as it returns to the last point it left off before the switch. From this point on all of the points generated by the source code are offset 51mm, allowing the second tool to continue tracing over the design made by the first tool.

Section III: MATLAB Source Code with Comments

```
% MATLAB code to generate G-code to be input to the food printer to print a
% hypocycloid structure. Note that to run this code, MATLAB R2016b or
% later is required to support local function calling.
%
% Course: Digital Manufacturing - Dr. Hod Lipson
% Creators: Jacob Joseph, Rayal Raj Prasad, Brian Jin, Jia Huang
% Department of Mechanical Engineering, Columbia University in the City of New York

close all
clear
clc

%Settings for print
x0 = 100;
y0 = 75;
R = 60/1.75;          %radius of large circle
r = 27.5/1.75;        %radius of small circle
a = 25/1.75;          %distance from center of small circle
layers = 20;
walls = 3;
ratio = .925;
z_height = 1.1;

%Counters
f = 0; %counter for # of hypocycloid curves printed
count = 0; %counter for # of walls

%Generate points for hypocycloid structure
P_final = [];
for i=1:layers
    for j = 1:walls
        if mod(f,2)
            P = epicycloid(x0,y0,(i-1)*z_height,ratio^count,R,r,a);
        else
            P = flip(epicycloid(x0,y0,(i-1)*z_height,ratio^count,R,r,a));
        end
        f = f + 1;
        count = count + 1;
        P_final = [P_final;P];
    end
    count = 0;
end

gauge = 1.29;
syringe = 16;

%Dual Extrusion
length = 0;                        %length extruded
vol_syringe = syringe^2/4*pi*70;    %approx volume of syringe

%switch to next syringe if 1 syringe runs out
for i = 2:size(P_final,1)

    length = length + pdist([P_final(i-1,1:3);P_final(i,1:3)],'Euclidean');
    ext_volume = gauge*length;      %approx volume extruded

    %if extruded volume is greater than 80% of syringe volume, insert a
    %flag into the P_final matrix indicating that the printer should switch
    %to the other extruder. The flag is only inserted after a layer is
    %printed
    if ext_volume > vol_syringe*.5
        row = i - mod(i,3000*walls);
        P_final(1:row-1,:) = P_final(1:row-1,:);
        tp = P_final(row:end,:)...
            - repmat([51 0 0 0],size(P_final(row:end,:),1),1);
        P_final(row,:) = [0 0 0 0];
        P_final(row+1:end+1,:) = tp;
        break

    end

end

% plotting required G-Code path
figure
plot3(P_final(:,1),P_final(:,2),P_final(:,3))

% call function to write G-Code file
writetogcode(P_final,gauge,syringe);

function writetogcode(V,gauge,syringe)
% This funtion takes an input vector, the gauge of the exturder and the
% cross sectional area of the extrusion chamber and outputs a gcode file
% from the vector of points. It also contains the tool intilization and
%coordinate and unit assigment as well as the priming path
%This version is used for mulitple tool changes Back and forth

in = 1;
fid = fopen('spirograph.gcode','w');      %open gcode file
fprintf(fid,'M106 S200 \n');              %turn on fan
fprintf(fid,'T0 \n');                     %Select Tool
fprintf(fid,'G90 \n');                    %absolute coordinate system
fprintf(fid,'G21 \n');                    %Set to millimeters
fprintf(fid,'G92 X0 Y0 Z0 E0 \n');        %set current position to zero
fprintf(fid,'G0 X0 Y0 Z0 E%f \n',in);     %move head to zero position

L = V(:,4);                               %Distance traveled between points
l = 50;                                   %Priming Line
ratio = (gauge^2)/syringe^2;              %Extrusion Ratio
E1 = 2*in + ratio*l;                      %Priming extrusion rate
E = extrusion(V,gauge,syringe,L);         %Extrusion values at each point
                                          %based on ratio and distance
%Priming path
Xstart = V(1,1)-l;
Ystart = V(1,2);
Zstart = V(1,3);
fprintf(fid,'G01 X%f Y%f Z%f E%f F400 \n',Xstart,Ystart,Zstart,2*in);

Xend = V(1,1);
Yend = V(1,2);
Zend = V(1,3);
fprintf(fid,'G01 X%f Y%f Z%f E%f \n',Xend,Yend,Zend,E1);


E2 = E1+1;
flag = 0;
%Design
for i = 2:size(V,1)

    if V(i,1) == 0 && V(i,2) == 0 && V(i,3) == 0 && V(i,4) == 0

        fprintf(fid,'G0 X0 Y0 Z%f E%f \n',V((i-1),3),E(i)+E1);
        fprintf(fid,'T1 \n');
        fprintf(fid,'G01 X%f Y%f Z%f E%f F400 \n',...
                Xstart,Ystart,V((i-1),3),E(i-1)+E1);
        fprintf(fid,'G01 X%f Y%f Z%f E%f \n',...
                Xend-70,Yend,V((i-1),3),E(i-1)+E2);
        flag = 1;
    else
        X = V(i,1);
        Y = V(i,2);
        Z = V(i,3);
        if flag == 0
            fprintf(fid,'G01 X%f Y%f Z%f E%f \n',X,Y,Z,E(i)+E1);
        elseif flag == 1
            fprintf(fid,'G01 X%f Y%f Z%f E%f \n',X,Y,Z,E(i)+E2);
        end
    end

end

fprintf(fid,'G01 X%f Y%f Z%f E0 \n',Xstart,Ystart,V(end,3));
fprintf(fid,'M84 \n');
fclose(fid);
end



function [P] = RotZTheta(point,theta,center)    % function to rotate points

P0 = point - center;    % vector from point to center

% rotation matrix for rotation about z-axis
R = [cos(theta) -sin(theta) 0; sin(theta) cos(theta) 0; 0 0 1];

% rotation + translation
P = R*P0+center;

end

function E = extrusion(V,gauge,syringe,Length) % function to compute E

ratio = (gauge^2)/syringe^2; % extrusion ratio

E = zeros(1,size(V,1)); % empty matrix to store E values

% running through matrix to compute E using length and ratio
for i = 2:size(V,1)
    e = ratio.*Length(i);
    E(i) = e+E(i-1);  % incrementing to use with absolute coordinates
end
end

%function to produce epicycloid/hypocycloid curve
function P = epicycloid(x0,y0,z0,scale,R_,r_,a)

r = r_ * scale; %radius of smaller circle
R = R_ * scale; %radius of larger circle

k = R/r;
t = linspace(0,3*pi,3000);
jj = 0;

%Epicycloid Equations
% for i = 1:size(t,2)
%     jj = jj+1;
%     x(jj) = r*(k+1)*cos(t(i))-r*cos((k+1)*t(i));
%     y(jj) = r*(k+1)*sin(t(i))-r*sin((k+1)*t(i));
% end

%Hypocycloid Equations
for i = 1:size(t,2)
    jj = jj+1;
    x(jj) = r*(k-1)*cos(k*t(i))+a*cos((k-1)*t(i));
    y(jj) = r*(k-1)*sin(k*t(i))-a*sin((k-1)*t(i));
end

P = [x', y'];
```

```
P(:,3) = z0;
P(1,4) = 0;

for i = 2:size(P,1)
    P(i,4) = pdist([P(i,1:3);P(i-1,1:3)],'Euclidean');
end

%Rotate the shape; avoids interfering with priming line
P(:,1:3) = (rotz(-240)*P(:,1:3)')';

P(:,1) = P(:,1) + x0;
P(:,2) = P(:,2) + y0;

end
```
Published with MATLAB® R2016a

Section IV: G-Code

```
M106 S200
T0
G90
G21
G92 X0 Y0 Z0 E0
G0 X0 Y0 Z0 E1.000000
G01 X22.139472 Y58.210441 Z0.000000 E2.000000 F400
G01 X72.139472 Y58.210441 Z0.000000 E2.325020
G01 X72.080912 Y58.261584 Z0.000000 E2.325523
G01 X72.023224 Y58.313337 Z0.000000 E2.326026
G01 X71.966412 Y58.365694 Z0.000000 E2.326526
G01 X71.910479 Y58.418652 Z0.000000 E2.327025
G01 X71.855430 Y58.472206 Z0.000000 E2.327523
G01 X71.801266 Y58.526354 Z0.000000 E2.328020
G01 X71.747991 Y58.581089 Z0.000000 E2.328515
G01 X71.695609 Y58.636409 Z0.000000 E2.329009
G01 X71.644123 Y58.692308 Z0.000000 E2.329502
G01 X71.593535 Y58.748783 Z0.000000 E2.329994
G01 X71.543848 Y58.805830 Z0.000000 E2.330484
G01 X71.495067 Y58.863444 Z0.000000 E2.330974
G01 X71.447194 Y58.921620 Z0.000000 E2.331463
G01 X71.400231 Y58.980355 Z0.000000 E2.331951
G01 X71.354182 Y59.039644 Z0.000000 E2.332438
G01 X71.309050 Y59.099483 Z0.000000 E2.332925
G01 X71.264837 Y59.159867 Z0.000000 E2.333411
G01 X71.221547 Y59.220792 Z0.000000 E2.333896
G01 X71.179182 Y59.282254 Z0.000000 E2.334380
G01 X71.137744 Y59.344247 Z0.000000 E2.334865
G01 X71.097238 Y59.406768 Z0.000000 E2.335349
G01 X71.057665 Y59.469812 Z0.000000 E2.335832
G01 X71.019027 Y59.533374 Z0.000000 E2.336315
G01 X70.981329 Y59.597450 Z0.000000 E2.336798
G01 X70.944571 Y59.662035 Z0.000000 E2.337281
G01 X70.908757 Y59.727125 Z0.000000 E2.337764
G01 X70.873890 Y59.792714 Z0.000000 E2.338247
```

```
G01 X70.839971 Y59.858800 Z0.000000 E2.338730
G01 X70.807003 Y59.925375 Z0.000000 E2.339213
G01 X70.774988 Y59.992437 Z0.000000 E2.339696
G01 X70.743929 Y60.059980 Z0.000000 E2.340180
G01 X70.713829 Y60.128000 Z0.000000 E2.340664
G01 X70.684688 Y60.196492 Z0.000000 E2.341148
G01 X70.656510 Y60.265450 Z0.000000 E2.341633
G01 X70.629297 Y60.334871 Z0.000000 E2.342118
G01 X70.603051 Y60.404749 Z0.000000 E2.342604
G01 X70.577773 Y60.475080 Z0.000000 E2.343090
G01 X70.553467 Y60.545859 Z0.000000 E2.343577
G01 X70.530133 Y60.617081 Z0.000000 E2.344065
G01 X70.507775 Y60.688740 Z0.000000 E2.344554
G01 X70.486393 Y60.760833 Z0.000000 E2.345044
G01 X70.465990 Y60.833355 Z0.000000 E2.345534
G01 X70.446568 Y60.906299 Z0.000000 E2.346026
G01 X70.428129 Y60.979662 Z0.000000 E2.346519
G01 X70.410673 Y61.053438 Z0.000000 E2.347013
G01 X70.394204 Y61.127623 Z0.000000 E2.347508
G01 X70.378722 Y61.202212 Z0.000000 E2.348005
G01 X70.364230 Y61.277198 Z0.000000 E2.348502
G01 X70.350729 Y61.352579 Z0.000000 E2.349002
G01 X70.338220 Y61.428347 Z0.000000 E2.349502
G01 X70.326706 Y61.504499 Z0.000000 E2.350004
G01 X70.316187 Y61.581030 Z0.000000 E2.350508
G01 X70.306665 Y61.657933 Z0.000000 E2.351013
G01 X70.298141 Y61.735205 Z0.000000 E2.351520
G01 X70.290617 Y61.812839 Z0.000000 E2.352029
G01 X70.284094 Y61.890832 Z0.000000 E2.352540
G01 X70.278574 Y61.969177 Z0.000000 E2.353052
G01 X70.274058 Y62.047870 Z0.000000 E2.353566
G01 X70.270546 Y62.126906 Z0.000000 E2.354083
G01 X70.268040 Y62.206278 Z0.000000 E2.354601
G01 X70.266541 Y62.285983 Z0.000000 E2.355121
G01 X70.266051 Y62.366015 Z0.000000 E2.355643…
```

Section V: Path Simulated

The G-Code was opened with Repetier. Fig. 14, 15 and 16 shows the path simulated.

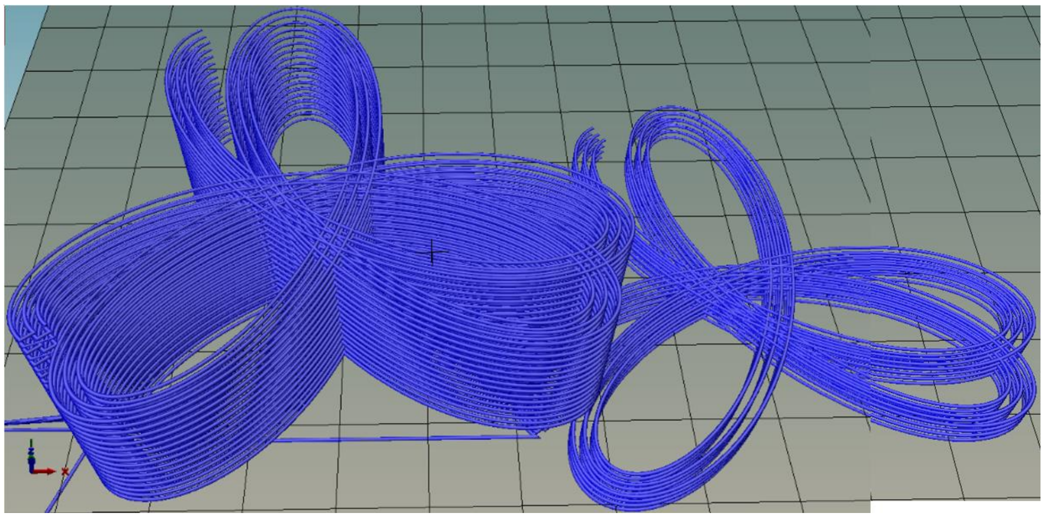

Figure 14: Simulated Path - I

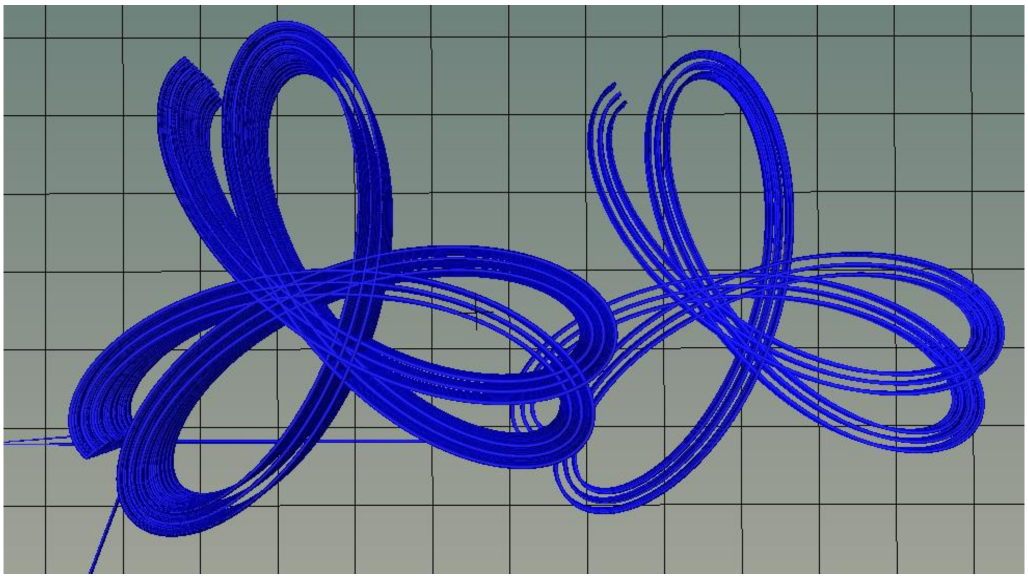

Figure 15: Simulated Path - II

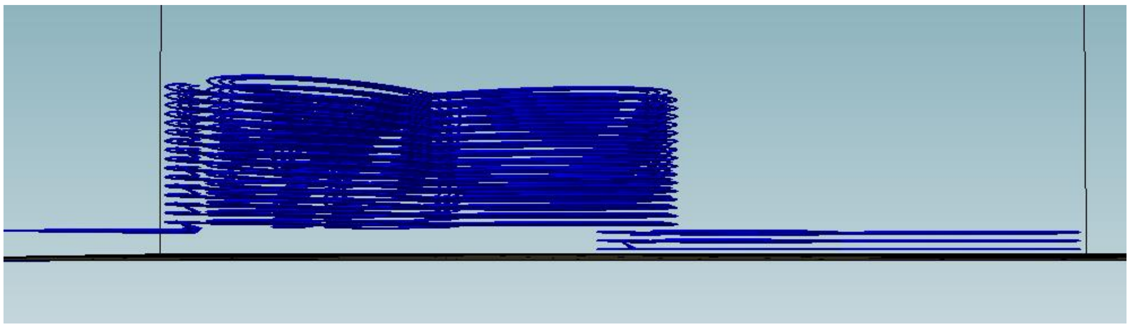

Figure 16: Simulated Path – III

Section VI: Problems Encountered, Solutions and Documentation

Material Selection

Several problems were encountered when it came to material selection. Semolina flour was initially used with the intent to make a ravioli style pasta. The first batch did not contain enough flour and the batch was too watery (Fig. 17). The dough was not able to cleanly stack on top of the previous layer. The next iteration involved adding more flour into the dough mixture to increase its consistency. The resulting dough was more viscous and shapeable. When this new dough mixture was being printed, the flow was quite restricted but it was able to stack onto the first layer. However, shortly after the print began the stepper motor on the extruder stalled. The consistency of the dough was too thick and caused too much friction between the material and the wall. The extruder does not have the power to extrude a material of this consistency.

The next iteration in material section involved printing with tofu. Silken Tofu with a medium firmness was first battered in a cup, then filled into the syringe. However, the tofu itself contained too much water, and was not able to stack when printed. The material was also very clumpy, with a lot of air bubbles in the syringe. This resulted in a print with inconsistent texture and poor shape.

Our last iteration involved mixing the tofu and semolina flour together (Fig. 18). The flour absorbed the excess water in the tofu, resulting in a consistency similar to cream cheese. This mixture was prepared and battered in a cup, which was then filled into the syringe using a popsicle stick. Printing with this hybrid material produced very good results. The texture of the print was very uniform, and multiple layers can be stacked. This mixture was named To-Dough (Tofu and Dough), and was the final material used for this chapter of the project. It stacks very well when printed and has a semi-uniform texture. It was microwaved to a golden color, with a hard texture, seen in Fig. 19.

To-Dough was an ideal material for food printing. However, avoiding air bubbles during the syringe filling process was still difficult. To solve this problem, canola oil was added into the syringe prior to the filling. This was done by dipping the popsicle stick in canola oil, then rubbing it along the inside wall of the syringe. This coat of oil helps in the filling process by allowing the To-Dough to fall to the bottom of the syringe, pushing out the air bubbles that would otherwise be trapped at the bottom. The oil also reduced friction when the material is being forced out, creating a smoother print and reducing the number of air gaps created by the extruder.

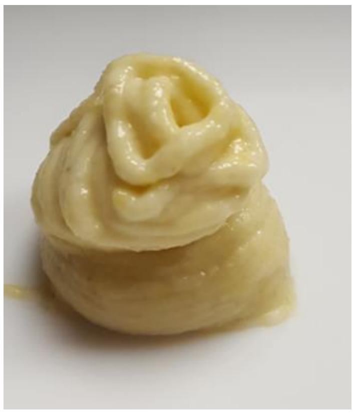
Figure 17: Dough Only

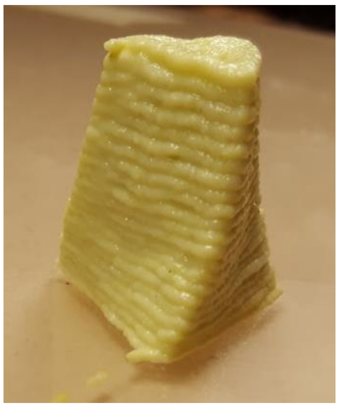
Figure 18: To-Dough (Tofu and Dough)

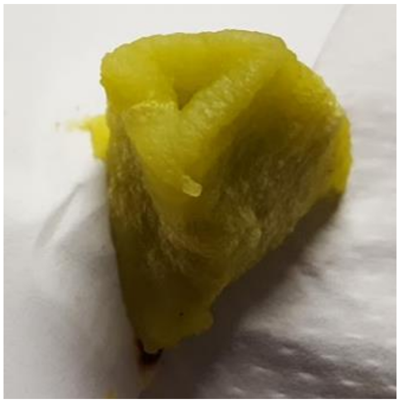
Figure 19: Microwaved To-Dough

Iteration 1

The first iteration, made of icing, was designed with 15 loops, R = 60mm, one wall, and a single layer shown in Fig. 20. Because of the number of loops there was not enough material in one syringe to add more walls and more layers to get a complete print. Additionally, a stop extrusion region was written into the code and placed near the center to limit the excess buildup of material being layered on top on each other. This feature proved problematic because the constant stop start of the extruder caused air pockets to form that caused excessive gaps in the final print. Moreover, to add layers and increase the height of the printed design a smaller object with less loops and more walls needed to be generated so that it would be stable and require only amount of material capable of being used by a syringe.

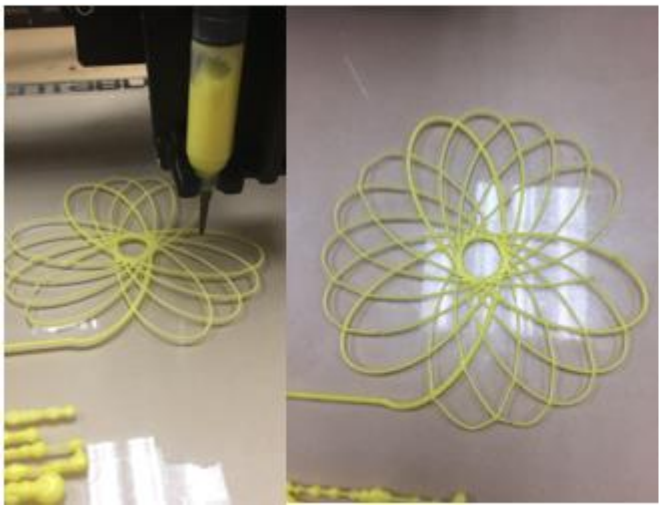

Figure 20: 15 loop iteration

Iteration 2

For iteration 2, the hypocycloid shape is produced with the To-Dough material shown in Fig. 21. The settings were set to 2 walls with R = 60, r = 27, and a = 25. The number of layers was set to 20 with a z-height of approximately 1.3 mm per layer. The main problem encountered was that the volume of the structure was larger than the volume of material contained in the syringe; consequently, only 4-5 layers could be printed. Additionally, 2 walls were not enough to support the load as the layers were built up, which caused the structure to sag. To improve the print, the size of the hypocycloid shape was reduced, the number of walls was increased, and the z-height between each layer would be reduced to adjust for sagging. Also, dual extrusion would need to be included so that double the syringe volume would be available for the print.

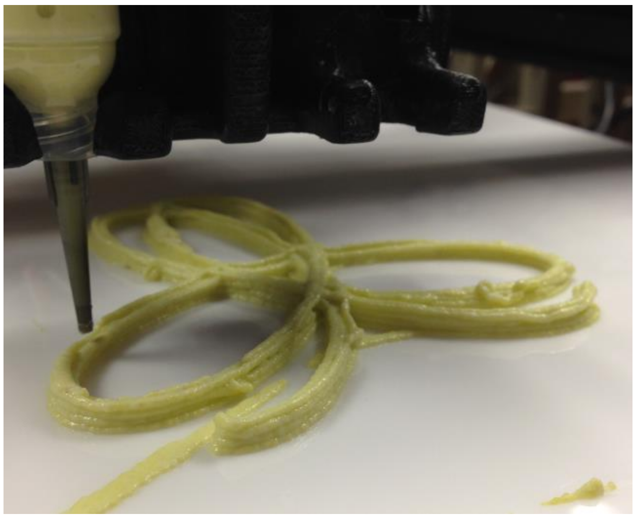

Figure 21:Large size, 4 layers

Iteration 3

For the iteration 3, the hypocycloid parameters were reduced to approximately 57% of the original values. An initial print with two walls and one extrusion tool is shown in Fig. 22. The walls began to sag and there was not enough material to increase the number of layers. For the final print (Fig. 23) the number of walls was also increased to 3 and the z - height between each layer was reduced to approximately 1.1 mm per layer. Dual extrusion was also included, and food coloring was added to the To-Dough material to indicate the switch from one syringe to the other. Blue indicates the first syringe and pink indicates the second syringe. There is a disproportionate volume of pink material compared to the blue material because the first syringe was approximately 50% full and the second syringe was almost 100% full. From the dual extrusion feature, we were able to achieve about 15 layers before both extruders ran out of material.

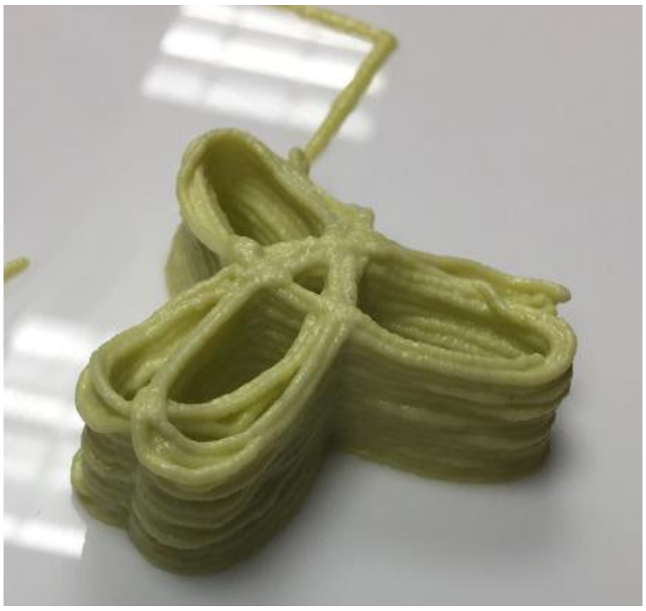
Figure 22: Small size, 13 layers

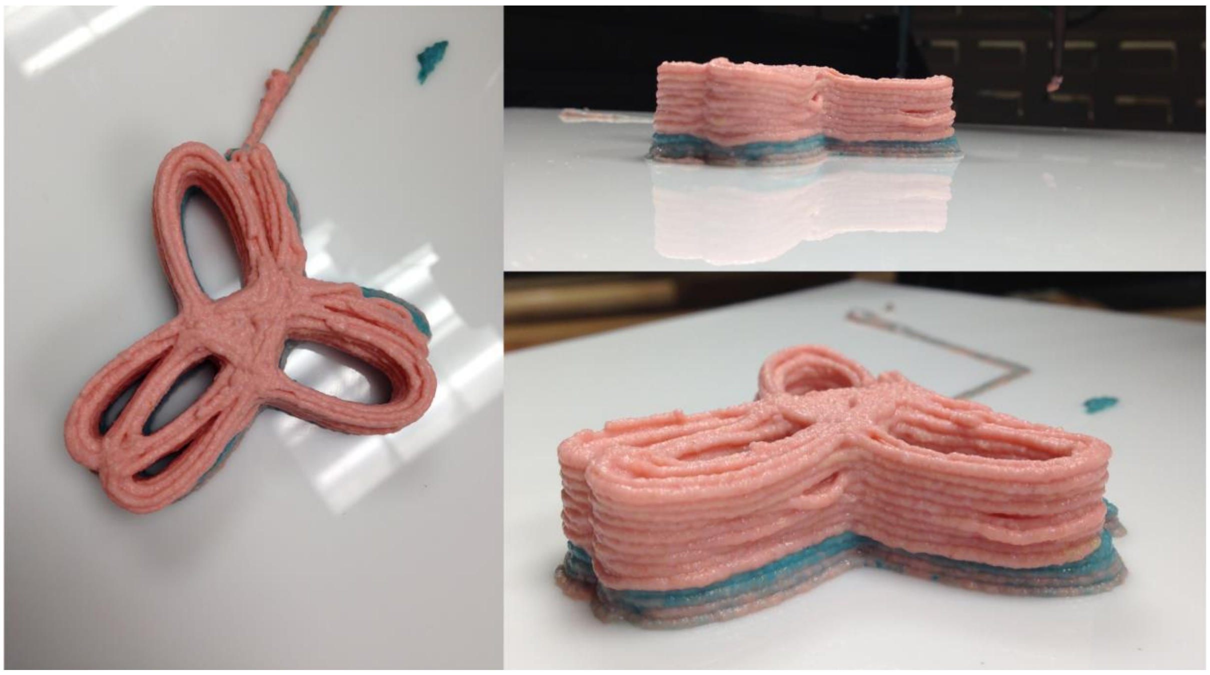
Figure 23: Small size, 15 layers

Section VII: Videos

There were many videos shot of the process through the iterations and uploaded to YouTube.

(recommend watching at a higher speed using video settings on YouTube)

Iteration 1 is demonstrated in the first link – Link (https://youtu.be/tgfNFIXMoZs)

Iteration 3 (Fig. 22) is demonstrated in the second link – Link (https://youtu.be/zDEtzU2LA9o)

Iteration 3 (Fig. 23) is demonstrated in the third link – Link (https://youtu.be/eiYOungqBFg)

Part IV: Flower with two materials

Section I: Description

Two materials were used to print a flower design made of avocado and Nutella infused dough. The dough was made with semolina flour from Bob's Red Mill. Additional leaves were added to the design to complete the flower and add an aesthetic touch to the plate. Food coloring from Club House was used to color the avocado as well as the flowers. The two materials are alternated every three layers, however any number of layers to alternate can be specified. In order to stack the design high, three walls were used and the size of the cross section was decreased parabolically as each layer was printed. Every switch between materials included a priming line. Additionally, a method for switching two empty syringes with full syringes in the middle of the print was implemented so that the size of the design would not be limited by the volume of two syringes. A new material loading method involving lubricating the syringes before loading them was implemented and proved very effective in reducing problems related to air pockets and extrusion blocks that were encountered during the previous prints.

Role Distribution

A simplified RACI matrix was formulated to easily explain the distribution of the roles within the team.

	Brainstorming	**Recipe Design & Purchase**	**Coding - Structure Design**	**Coding for G-Code Generation**	**Printer and material setup**	**Printing and Troubleshooting**
Brian Jin	R	P	R	P	R	R
Jacob Joseph	R	P	P	R	R	R
Rayal Raj Prasad	R	P	R	P	R	R
Jia Huang	R	R	P	P	R	R

Legend

P	Participating
R	Responsible
X	Neither participating nor responsible

Process Approach

Flower Design

The cross section of the design is shown in Fig. 24 with multiple walls. An offset with the size of the extruded material (approximated 1.29 mm for a 16-gauge syringe tip) is added to each layer and its walls in order to account for the thickness of material extruded. A loop to convert all of the points plotted in polar coordinates to Cartesian coordinates was written to create the vectors used to output the G-code for the machine.

The equations (in polar coordinates) below represent the circular shapes of the flower design of Fig. 25. The first equation representing the innermost shape and the ones following represent the shapes from in to out.

$$r_0 = 1$$

$$r_1 = 2 + \cos(6\theta + \pi)$$

$$r_2 = 4 + \cos(6\theta)$$

$$r_3 = 6 + \cos(6\theta + \pi)$$

$$r_4 = 8 + \cos(6\theta)$$

$$r_5 = 10 + \cos(6\theta + \pi)$$

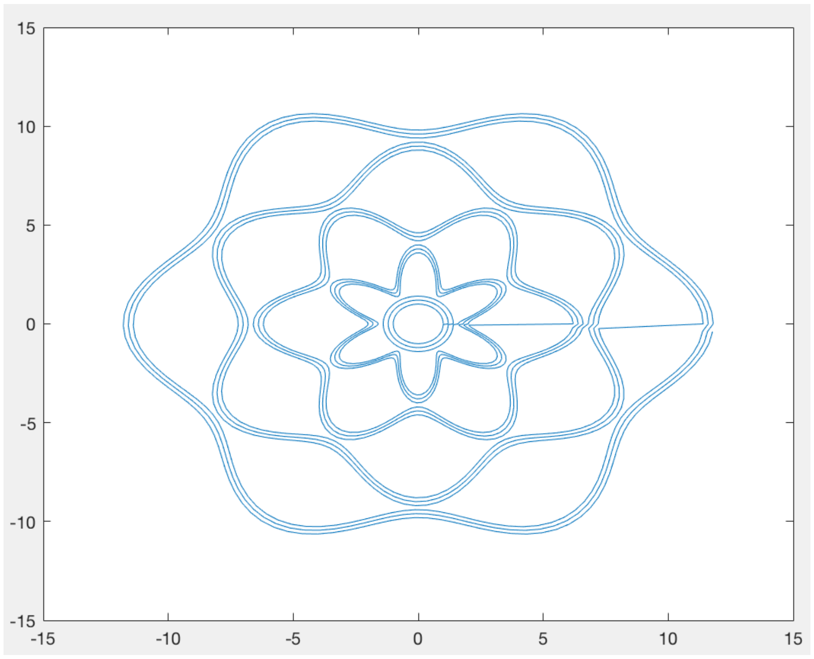

Figure 24: Cross-section of flower

The points from equations were plotted in polar coordinates and stored in a vector. These were then converted to Cartesian coordinates to be used to create the G-code. The following equations are used for the conversion:

$$x = r\cos(\theta)$$

$$y = r\sin(\theta)$$

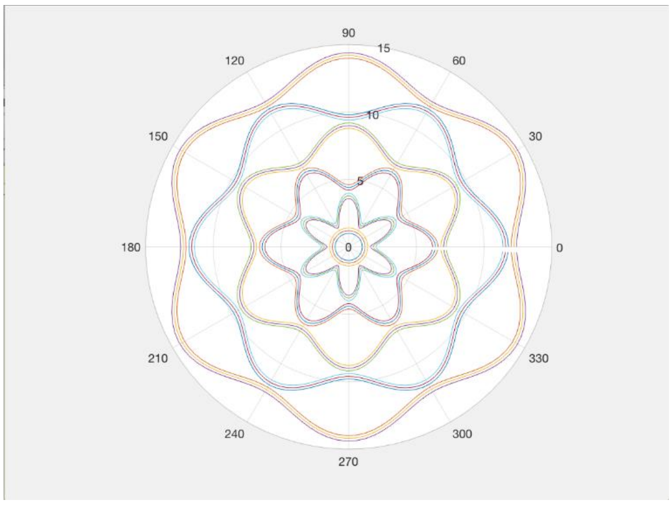

Figure 25: Polar plot of flower

Points from the first five equations were converted to Cartesian coordinates and the plot is shown in Fig. 24.

Leaf Design

The design of the leaf was created in a similar fashion to flower. A nonlinear function was plotted in polar coordinates as shown in Fig. 26. The same number of points were plotted per wall and an offset equal to the thickness of the extruder head for multiple walls was implemented. The following equation represents one leaf shape on a two dimensional plot. Again the points produced in polar coordinates were converted to Cartesian coordinates and written to G-code.

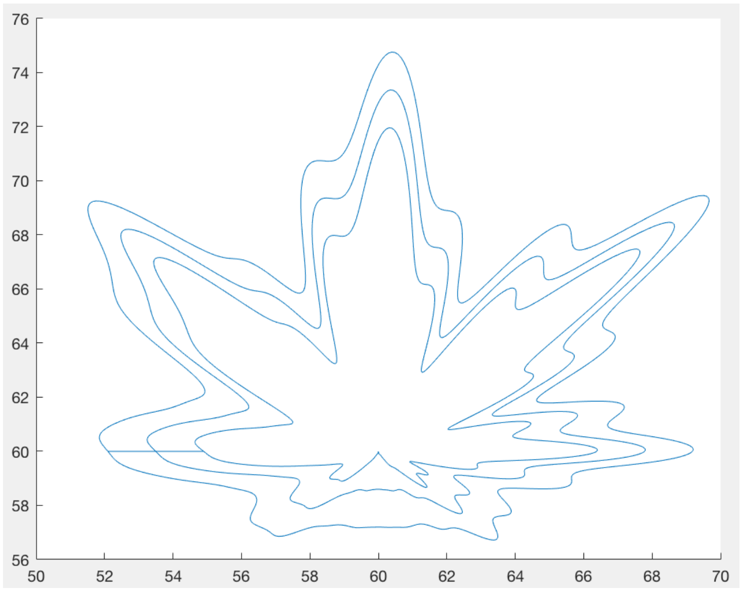

Figure 26: Cross section of Leaf

Equations for the leaf:

$$r = (leafThickness + 9\cos(8\theta))(1 + \cos(24\theta))(9 + \cos(numberOfPoints\ \theta))(1 + \sin(\theta))$$

Where *leafThickness* represents the width of each spike on the leaf (there are a total of 7 in Fig. 26). And *numberOfPoints* represents the overall number of sharp points or local maximums present. The polar plot of the leaf is shown in Fig. 27.

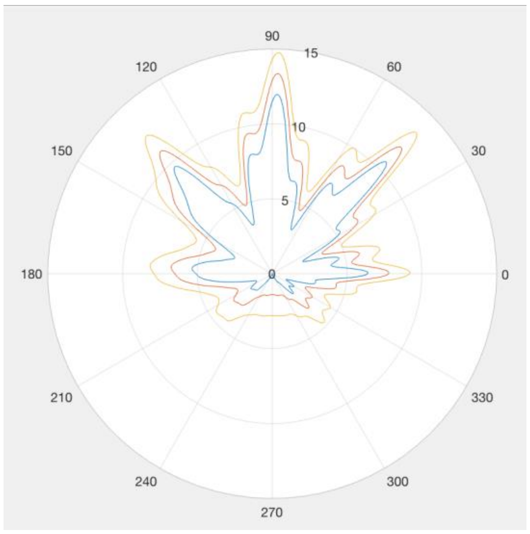

Figure 27: Polar plot of leaf

Alternating Materials

Dual extrusion was implemented in the structure and the material was switched every third layer. The code was robust to switching after any number of layers, with or without dual extrusion. After each switch, a new priming line was extruded to reduce the error from air pocket buildup once the extruder head reached the design.

The code also accounts for the offset between both nozzles, which can be seen in the simulation of the G-code. Although the initial data that was provided only had an offset in the X-axis of 50mm, the inconsistencies in the 3D printed bracket lead to the actual offset being 51 mm in the X-axis, and 0.8 mm in the Y-axis. Accounting for this offset was crucial to layer the material perfectly on top of one another.

Switching out Materials

The structure planned was 33 layers high, which required more than two syringes of material to print. To account for this, the G-code was split into two parts to be run before and after the syringes were switched out.

To determine the switch, the volume of the material extruded is estimated and compared to the total volume of the syringe as each point in the G-code is produced. This comparison is described by the equation:

$$V_{extruded} > 2\,Vol_{syringe}\,F_{safety}$$

A factor of safety of 80% was included to account for errors in the volume estimation.

Once the code computes where the switching of batches needed to occur, it rounds this layer down to the first layer at which a change of material takes place. This is done to ensure consistency with the print. If dual extrusion is not selected, the layer switch occurs exactly at the layer where the volume condition is met.

Switching the batch of syringes is a precise and time-sensitive task. It was critical to replace the syringes without moving the position of the extruder, which was achieved by running an intermediate G-code file that engages the stepper motors.

Section II: Algorithm

Main Script

The main script is divided into two sections: 1 that generates points for the flower and 1 that generates points for the leaf. Sections can be run individually or in succession. This script calls functions that plot the points for each design, converts them to Cartesian coordinates, and stores the X, Y, Z coordinates in a vector. Inputs that the user can define to change the print include: Overall size, number of layers, number of walls, syringe body diameter, syringe tip diameter (gauge), Material switching point, center point of print in the workspace, scaling of cross sectional area as height increases. The flower section also contains an additional feature that allows the user to choose to print with either one extruder or two extruders. For dual extrusion, the number of layers between each extruder may also be selected. The loop will automatically determine if 2 syringes or 4 syringes are necessary to complete the print according to the volume of the syringe. The main function stores coordinates for the design in a matrix and sends the matrix to the “writetogcode” functions. It also stores information about when to switch extruder heads and when to swap out syringes inside the coordinate matrix.

Write to G-code

A similar program to the previous parts but with a few improved features is written that outputs G-code from a vector produced by MATLAB code. It also calls and extrusion function that calculates the extrusion distance between each point by multiplying the distance between each point by the ratio of syringe diameter to extrusion tip diameter plus a tolerance factor. This extrusion vector is calculated and added to the vectors of X, Y, Z coordinates. Additionally, it looks for a flag (row of zeros) in the vector that plots all the points for the design. This flag represents the point when a tool switch occurs and a new material is used. When the flag is hit, the code switches the tools and implements a new priming line that also accounts for the offset between tools. A variable is used to keep track of the accumulated priming extrusion required for each tool switch and is added to the rest of the extrusion values for the duration of the print.

A modified output file was made for structures that couldn’t be printed within one batch on syringes. This new modified file “writetogcode_half.m” takes in an additional input which is the layer at which the extruders run out of material and need to be replaced. The code first rounds this layer number down to the layer at which the extruder switches, and then splits the over G-code into two files. An intermediate code to engage the stepper motors while switching the syringes was also incorporated to make the syringe switch robust to jerks.

Section III: MATLAB Source Code with Comments

```
% MATLAB code to generate G-code to be input to the food printer to print a
% flower and leaf structure. Note that to run this code, MATLAB R2016b or
% later is required to support local function calling.
%
% Course: Digital Manufacturing - Dr. Hod Lipson
% Creators: Jacob Joseph, Rayal Raj Prasad, Brian Jin, Jia Huang
% Department of Mechanical Engineering, Columbia University in the City of New York
```

Flower

```
close all
clear
global gauge
global syringe

%Constants
gauge = 1.29;
syringe = 16;
vol_syringe = syringe^2/4*pi*70;

%Print Settings
resolution = 1000;                   %number of points per revolution traced
walls = 3;                           %number of walls
offset = 1.4;                        %distance between each wall
scale = 20;                          %shape of flower
s = 0.65;                            %size of base cross section
center = [90 60];                    %position of center of flower
layers = 33;                         %number of layers
z_height = linspace(1.05,0.9,layers);    %height per layer
cutoffR = 3;                             %radius of center hole

%Dual Extrusion Setting
dualExtrusion = true;
alternate = 3;                       %switch syringe every "alternate" layer


%Function variables
Pflower_final = [];
length = 0;
layerSwitch = 0;
splitRow = 1;

%Booleans
extruder_flag = 0;
first2Syringes = 1;
findRow = 0;

for i=1:layers

    %Generate xyz points for the ith layer
    if mod(i,2)
        P = Flower(center,(i-1)*z_height(i),s,walls,-((i-
1)/scale)^2+2,offset,resolution,cutoffR);
    else
        P = flip(Flower(center,(i-1)*z_height(i),s,walls,-((i-
1)/scale)^2+2,offset,resolution,cutoffR));
    end

    %Calculate the distance between each xyz point, store length in P. Also
    %calculates the volume of material that has been extruded
    P(1,4) = 0;
    for j = 2:size(P,1)
        P(j,4) = pdist([P(j,1:3);P(j-1,1:3)],'Euclidean');
        length = length + P(j,4);
    end

    ext_volume = length*gauge;

    %Check to see if the volume extruded exceeds the volume of 2 syringes.
    %If the condition is met, set the first2Syringes boolean to 0, which
    %will indicate later to split the g-code into 2 files
    if ext_volume > 2*vol_syringe*.8 && first2Syringes
        layerSwitch = i - mod(i,alternate);
        first2Syringes = 0;
    end

    %if dual extrusion is selected, then offset x coordinates by 51 mm
    if(extruder_flag==1 && dualExtrusion)
        P = P - repmat([51 -.8 0 0],size(P,1),1);
    end

    Pflower_final = [Pflower_final;P];

    %insert row of zeros into Pflower_final after every "alternate" layers.
    %A row of zeros tells the writetogcode function when to switch
    %extruders.
    if (mod(i,alternate)==0 && dualExtrusion)
        Pflower_final = [Pflower_final;zeros(1,size(Pflower_final,2))];
        extruder_flag = 1-extruder_flag;
    end

    %Finds row in Pflower_final matrix where the g-code should be split
    %into 2 files. Only runs if first2Syringes is 0
    if first2Syringes == 0 && findRow == 0
        splitRow = size(Pflower_final,1);
        findRow = 1;
    end

end

%Plot the Pflower_final points
figure
plot3(Pflower_final(:,1),Pflower_final(:,2),Pflower_final(:,3))
axis equal
%zlim([25 35])

Pflower_final(1,4) = 0;

%Split the g-code if the first2Syringes is false
if first2Syringes
    writetogcode3(Pflower_final);
else
    writetogcode_half(Pflower_final,splitRow,dualExtrusion)
end
```

Leaf

```
close all
clear all
global gauge
global syringe

%Constants
gauge = 1.29;
syringe = 16;


%Print Settings
resolution = 1000;                      %number of points per layer
walls = 2;                              %number of walls
```

```
offset = 2.5;                           %distance between walls

s = 7;                                  %size of leaf
center = [60 60];                       %center position
z_height = 1.1;                         %height per layer
layers = 4;                             %number of layers

Pleaf_final = [];
for i=1:layers

    %Generate xyz points for the ith layer
    if mod(i,2)
        P = Leaf1(center,(i-1)*z_height,s,walls,1,offset,resolution);
    else
        P = flip(Leaf1(center,(i-1)*z_height,s,walls,1,offset,resolution));
    end

    Pleaf_final = [Pleaf_final;P];

end

%plot Pleaf_final points
figure
plot3(Pleaf_final(:,1),Pleaf_final(:,2),Pleaf_final(:,3))

Pleaf_final(1,4) = 0;

%Calculate the distance between each point in Pleaf_final
for i = 2:size(Pleaf_final,1)
    Pleaf_final(i,4) = pdist([Pleaf_final(i,1:3);Pleaf_final(i-
1,1:3)],'Euclidean');
end

writetogcode_leaf(Pleaf_final)

function [P] = Leaf1(center,z,s,walls,scale,offset,resolution)

%Leaf shape parameters
theta = linspace(-pi,pi,resolution);
leafThickness = 2;
numberOfPoints = 25;


R = zeros(walls,size(theta,2));

%Leaf equation
R(1,:) =
scale*s*(leafThickness+.9*cos(8*theta)).*(1+.1*cos(24*theta)).*(.9+.1*cos(numberOfP
oints*theta)).*(1+sin(theta));

for i = 2:walls
    R(i,:) = R(i-1,:)+offset;
end

%plot leaf in polar coordinates
polarplot(theta,R)

%convert from polar coordinates to Cartesian coordinates
P_prime = zeros(size(R,1)*size(R,2),2);
row = 1;
for i = 1:size(R,1)
    for j = 1:size(R,2)
        P_prime(row,:) = [R(i,j)*cos(theta(j)) R(i,j)*sin(theta(j))];
        row = row + 1;
    end
end
```

```
P_prime = P_prime + repmat(center,size(P_prime,1),1);
P = [P_prime, repmat(z,size(P_prime,1),1)];     %add z coordinate to P

end

function [P] = Flower(center,z,s,walls,scale,offset,resolution,minR)

theta = linspace(0,2*pi,resolution);

%r0-7 equation for each circular shape in flower
r0=1*scale*s;
r1=offset*walls+(2+cos(theta*6+pi))*scale*s;
r2=2*offset*walls+(4+cos(theta*6))*scale*s;
r3=3*offset*walls+(6+cos(theta*6+pi))*scale*s;
r4=4*offset*walls+(8+cos(theta*6))*scale*s;

%vectors for each circular shape, each row represents a wall
r_0 = zeros(walls,size(theta,2));
r_0(1,:) = r0;
r_1 = zeros(size(r_0));
r_1(1,:) = r1;
r_2 = zeros(size(r_0));
r_2(1,:) = r2;
r_3 = zeros(size(r_0));
r_3(1,:) = r3;
r_4 = zeros(size(r_0));
r_4(1,:) = r4;

for i = 2:walls
    r_0(i,:) = r_0(i-1,:)+offset;
    r_1(i,:) = r_1(i-1,:)+offset;
    r_2(i,:) = r_2(i-1,:)+offset;
    r_3(i,:) = r_3(i-1,:)+offset;
    r_4(i,:) = r_4(i-1,:)+offset;

end

%plot the points in polar coordinates
polarplot(theta,r_0,theta,r_1,theta,r_2,theta,r_3,theta,r_4)

R = [r_0;r_1;r_2;r_3];      %consolidate r's into 1 matrix

%Checks to make sure no point is within an area defined by user-specified
%circle. This check ensures that the flower cross section is not too small
%for the printe resolution

deleteRow = 0;
tooSmall = false;
for i = 1:size(R,1)

    if min(R(end,:)) < minR
        tooSmall = true;
        break
    end

    if min(R(i,:)) > minR
        deleteRow = i;
        break
    end
end

if deleteRow ~= 0
    R = R(deleteRow:end,:);
end

if tooSmall
    R = zeros(1,size(R,1));
```

```
end

%Convert polar coordinates to Cartesian coordinates
P_prime = zeros(size(R,1)*size(R,2),2);
row = 1;
for i = 1:size(R,1)
    for j = 1:size(R,2)
        P_prime(row,:) = [R(i,j)*cos(theta(j)) R(i,j)*sin(theta(j))];
        row = row + 1;
    end
end

P_prime = [P_prime, repmat(z,size(P_prime,1),1)];   %add z coordinates to P_prime
P_prime(:,1:3) = (rotz(180)*(P_prime(:,1:3)'))';    %rotate coordinates so to
prevent interference with priming line

P = P_prime + repmat([center 0],size(P_prime,1),1);

end

function E = extrusion(L)
global gauge
global syringe
ratio = (gauge^2)/syringe^2;
% e = ratio.*Length;
flag = 0;
E = zeros(1,size(L,1));
for i = 2:size(L,1)
    e = ratio.*L(i);
%     if((V(i,1)-x0)^2 + (V(i,2)-y0)^2) <= (R/4)^2
%         E(i) = E(i-1);
%         flag = 1;
%     else
        if flag == 1
            E(i) = 3*e+E(i-1);
        else
            E(i) = e+E(i-1);
        end
        flag = 0;

%    end
end
end

function writetogcode3(V)
global gauge    % gauge and syringe body diameter are defined as global variables
global syringe
initialE = 1;   % initial extrusion value used to initialize priming

fid = fopen('flower.gcode','w');    % creating new gcode file for the flower
fprintf(fid,'M106 S200 \n');    %turn on fan
fprintf(fid,'T0 \n');
fprintf(fid,'G90 \n');  %absolute coordinate system
fprintf(fid,'G21 \n');  %Set to millimeters
fprintf(fid,'G92 X0 Y0 Z0 E0 \n');  %set current position to zero
fprintf(fid,'G0 X0 Y0 Z0 E0 \n');

L = V(:,4); % extracting the Euclidean distances between points

l = 50; % length of the priming line

ratio = (gauge^2)/syringe^2;
E1 = (initialE + ratio*l);  % counter variable used if only a single extruder is
used
E = extrusion(L);   % function to compute extruder values between every point

% horizontal priming path of length 'l'
```

```
Xstart = V(1,1)-l;
Ystart = V(1,2);
Zstart = V(1,3);
fprintf(fid,'G01 X%f Y%f Z%f E%f F400 \n',Xstart,Ystart,Zstart,initialE);

Xend = V(1,1);
Yend = V(1,2);
Zend = V(1,3);
fprintf(fid,'G01 X%f Y%f Z%f E%f \n',Xend,Yend,Zend,E1);

% flags
T0 = 1; % flag used during dual extrusion tool change
flag = 0;   % flag to indicate if a row of zeroes is hit
E2 = 1; % counter variable that accumulates the value of the extruder for dual
extrusion

% design
for i = 2:size(V,1) % loop runs from the second point

    % encountering a row of zeroes i.e. change extruder
    if V(i,1) == 0 && V(i,2) == 0 && V(i,3) == 0 && V(i,4) == 0

        % toggling tool flag
        T0 = (1-T0);

        % writing points for the extruder
        if flag == 1
            fprintf(fid,'G0 X0 Y0 Z%f E%f \n',V((i-1),3),E(i)+E2);
        else
            fprintf(fid,'G0 X0 Y0 Z%f E%f \n',V((i-1),3),E(i)+E1);
        end

        % depending on the extruder used, the offset is either included or not
        if T0
            fprintf(fid,'T0 \n');
            if flag == 1
                fprintf(fid,'G01 X%f Y%f Z%f E%f F400 \n',X-l,Y,V((i-1),3),E(i-
1)+E2);
            else
                fprintf(fid,'G01 X%f Y%f Z%f E%f F400 \n',X-l,Y,V((i-1),3),E(i-
1)+E1);
            end
            E2 = E2+E1;
            fprintf(fid,'G01 X%f Y%f Z%f E%f \n',X,Y,V((i-1),3),E(i-1)+E2);
        else
            fprintf(fid,'T1 \n');
            if flag == 1
                fprintf(fid,'G01 X%f Y%f Z%f E%f F400 \n',X-2*l,Y,V((i-1),3),E(i-
1)+E2);

            else
                fprintf(fid,'G01 X%f Y%f Z%f E%f F400 \n',X-2*l,Y,V((i-1),3),E(i-
1)+E1);
            end
            E2 = E2+E1;
            fprintf(fid,'G01 X%f Y%f Z%f E%f \n',X-51,Y,V((i-1),3),E(i-1)+E2);
        end

        % flag set to 1 to indicate dual extrusion - will need different extrusion
values
        flag = 1;
    % logic comes here if it is a non-zero row
    else
        X = V(i,1);
        Y = V(i,2);
        Z = V(i,3);
        % corresponding statements depending on whether dual extrusion is active or
not
        if flag == 1
            fprintf(fid,'G01 X%f Y%f Z%f E%f \n',X,Y,Z,E(i)+E2);
        else
            fprintf(fid,'G01 X%f Y%f Z%f E%f \n',X,Y,Z,E(i)+E1);
        end
    end
end

% printing the end of the gcode file
fprintf(fid,'G01 X%f Y%f Z%f E0 \n',Xstart,Ystart,V(end,3));    % go back to home
position
fprintf(fid,'M84 \n');  % disable motors
fclose(fid);    % close file
end

function writetogcode_leaf(V)
global gauge    % gauge and syringe body diameter are defined as global variables
global syringe
initialE = 1;   % initial extrusion value used to initialize priming
fid = fopen('leaf.gcode','w');  % creating new gcode file for the leaf
fprintf(fid,'M106 S200 \n');    % turn on fan
fprintf(fid,'T0 \n');
fprintf(fid,'G90 \n');  % absolute coordinate system
fprintf(fid,'G21 \n');  % set to millimeters
fprintf(fid,'G92 X0 Y0 Z0 E0 \n');  % set current position to zero
fprintf(fid,'G0 X0 Y0 Z0 E0 \n');

L = V(:,4); % extracting the Euclidean distances between points

l = 50; % length of the priming line

ratio = (gauge^2)/syringe^2;
E1 = (initialE + ratio*l);  % counter variable used if only a single extruder is
used
E = extrusion(L);   % function to compute extruder values between every point

% horizontal priming path of length 'l'
Xstart = V(1,1)-l;
Ystart = V(1,2);
Zstart = V(1,3);
fprintf(fid,'G01 X%f Y%f Z%f E%f F400 \n',Xstart,Ystart,Zstart,initialE);

Xend = V(1,1);
Yend = V(1,2);
Zend = V(1,3);
fprintf(fid,'G01 X%f Y%f Z%f E%f \n',Xend,Yend,Zend,E1);

% flags
T0 = 1; % flag used during dual extrusion tool change
flag = 0;   % flag to indicate if a row of zeroes is hit
E2 = 1; % counter variable that accumulates the value of the extruder for dual
extrusion

% design
for i = 2:size(V,1) % loop runs from the second point

    % encountering a row of zeroes i.e. change extruder
    if V(i,1) == 0 && V(i,2) == 0 && V(i,3) == 0 && V(i,4) == 0

        % toggling tool flag
        T0 = (1-T0);

        % writing points for the extruder
        if flag == 1
            fprintf(fid,'G0 X0 Y0 Z%f E%f \n',V((i-1),3),E(i)+E2);
        else
            fprintf(fid,'G0 X0 Y0 Z%f E%f \n',V((i-1),3),E(i)+E1);
        end

        % depending on the extruder used, the offset is either included or not
        if T0
            fprintf(fid,'T0 \n');
            if flag == 1
                fprintf(fid,'G01 X%f Y%f Z%f E%f F400 \n',X-l,Y,V((i-1),3),E(i-
1)+E2);
            else
                fprintf(fid,'G01 X%f Y%f Z%f E%f F400 \n',X-l,Y,V((i-1),3),E(i-
1)+E1);
            end
            E2 = E2+E1;
            fprintf(fid,'G01 X%f Y%f Z%f E%f \n',X,Y,V((i-1),3),E(i-1)+E2);
        else
            fprintf(fid,'T1 \n');
            if flag == 1
                fprintf(fid,'G01 X%f Y%f Z%f E%f F400 \n',X-2*l,Y,V((i-1),3),E(i-
1)+E2);

            else
                fprintf(fid,'G01 X%f Y%f Z%f E%f F400 \n',X-2*l,Y,V((i-1),3),E(i-
1)+E1);
            end
            E2 = E2+E1;
            fprintf(fid,'G01 X%f Y%f Z%f E%f \n',X-5l,Y,V((i-1),3),E(i-1)+E2);
        end

        % flag set to 1 to indicate dual extrusion - will need different extrusion
values
        flag = 1;
    % logic comes here if it is a non-zero row
    else
        X = V(i,1);
        Y = V(i,2);
        Z = V(i,3);

        % corresponding statements depending on whether dual extrusion is active or
not
        if flag == 1
            fprintf(fid,'G01 X%f Y%f Z%f E%f \n',X,Y,Z,E(i)+E2);
        else
            fprintf(fid,'G01 X%f Y%f Z%f E%f \n',X,Y,Z,E(i)+E1);
        end
    end
end

% printing the end of the gcode file
fprintf(fid,'G01 X%f Y%f Z%f E0 \n',Xstart,Ystart,V(end,3));    % go back to home
position
fprintf(fid,'M84 \n');  % disable motors
fclose(fid);    % close file
end

function writetogcode_half(V,split_row,dualExtrusion)
global gauge    % gauge and syringe body diameter are defined as global variables
global syringe
length = size(V,1); % variable to store the number of points in the structure

% find nearest row of zeros (round down)
while(split_row>0 && dualExtrusion)
    if(sum(V(split_row,:))~= 0)
        split_row = split_row - 1;
    else
        break
    end
end
split_row = split_row - 1;  % subtract 1 to get the row before extruder switch

initialE = 1;   % initial extrusion value used to initialize priming
fid1 = fopen('flower_part1.gcode','w'); % flower code - part 1
fid2 = fopen('flower_part2.gcode','w'); % flower code - part 2

L = V(:,4);

% writing file 1
fprintf(fid1,'M106 S200 \n');                         %turn on fan
fprintf(fid1,'T0 \n');
fprintf(fid1,'G90 \n');                               %absolute coordinate system
fprintf(fid1,'G21 \n');                               %Set to millimeters
fprintf(fid1,'G92 X0 Y0 Z0 E0 \n');                   %set current position to zero
fprintf(fid1,'G0 X0 Y0 Z0 E0 \n');

l = 70; % priming line longer to ensure extruders are outside print while switching
ratio = (gauge^2)/syringe^2;
E1 = (initialE + ratio*l); % storing extruder values
E2 = 1;

E = extrusion(L);

% priming path
Xstart = V(1,1)-l;
Ystart = V(1,2);
Zstart = V(1,3);
fprintf(fid1,'G01 X%f Y%f Z%f E%f F400 \n',Xstart,Ystart,Zstart,initialE);


Xend = V(1,1);
Yend = V(1,2);
Zend = V(1,3);
fprintf(fid1,'G01 X%f Y%f Z%f E%f \n',Xend,Yend,Zend,E1);

% flags
T0 = 1;
flag = 0;
Eoffset = 0;

%design
for i = 2:size(V,1)

    if i == split_row
       %write head files
        fprintf(fid2,'M106 S200 \n');                        %turn on fan\
        fprintf(fid2,'G90 \n');                              %absolute coordinate
system
        fprintf(fid2,'G21 \n');                              %Set to millimeters
        %ensuring correct extrusion value for second file
        fprintf(fid2,'G92 X%f Y%f Z%f E%f \n',Xstart,Y_1,Z_1,E(i-
1)+E2);             %set current position to zero
    end

    if sum(V(i,:)) == 0

        T0 = (1-T0);
        if i<split_row
            if flag == 1
                fprintf(fid1,'G0 X0 Y0 Z%f E%f \n',V((i-1),3),E(i)+E2);
            else
                fprintf(fid1,'G0 X0 Y0 Z%f E%f \n',V((i-1),3),E(i)+E1);%-Eoffest
removed
            end
        else
```

```
            if flag == 1
                fprintf(fid2,'G0 X0 Y0 Z%f E%f \n',V((i-1),3),E(i)+E2);
            else
                fprintf(fid2,'G0 X0 Y0 Z%f E%f \n',V((i-1),3),E(i)+E1);%-Eoffest
removed
            end
        end

        if i<split_row
            if T0
                fprintf(fid1,'T0 \n');
                if flag == 1
                    fprintf(fid1,'G01 X%f Y%f Z%f E%f F400 \n',0,Y,V((i-1),3),E(i-
1)+E2); % was X-1
                else
                    fprintf(fid1,'G01 X%f Y%f Z%f E%f F400 \n',0,Y,V((i-1),3),E(i-
1)+E1);
                end
                E2 = E2+E1;
                fprintf(fid1,'G01 X%f Y%f Z%f E%f \n',X,Y,V((i-1),3),E(i-1)+E2);
            else
                fprintf(fid1,'T1 \n');
                if flag == 1
                    fprintf(fid1,'G01 X%f Y%f Z%f E%f F400 \n',0,Y,V((i-1),3),E(i-
1)+E2); % was X-2*l

                else
                    fprintf(fid1,'G01 X%f Y%f Z%f E%f F400 \n',0,Y,V((i-1),3),E(i-
1)+E1);
                end
                E2 = E2+E1;
                fprintf(fid1,'G01 X%f Y%f Z%f E%f \n',X-51,Y,V((i-1),3),E(i-1)+E2);
            end
        else
            if T0
                fprintf(fid2,'T0 \n');
                if flag == 1
                    fprintf(fid2,'G01 X%f Y%f Z%f E%f F400 \n',0,Y,V((i-1),3),E(i-
1)+E2);
                else
                    fprintf(fid2,'G01 X%f Y%f Z%f E%f F400 \n',0,Y,V((i-1),3),E(i-
1)+E1);
                end
                E2 = E2+E1;
                fprintf(fid2,'G01 X%f Y%f Z%f E%f \n',X,Y,V((i-1),3),E(i-1)+E2);
            else
                fprintf(fid2,'T1 \n');
                if flag == 1
                    fprintf(fid2,'G01 X%f Y%f Z%f E%f F400 \n',0,Y,V((i-1),3),E(i-
1)+E2);

                else
                    fprintf(fid2,'G01 X%f Y%f Z%f E%f F400 \n',0,Y,V((i-1),3),E(i-
1)+E1);
                end
                E2 = E2+E1;
                fprintf(fid2,'G01 X%f Y%f Z%f E%f \n',X-51,Y,V((i-1),3),E(i-1)+E2);
            end
        end
        flag = 1;
    else
        X = V(i,1);
        Y = V(i,2);
        Z = V(i,3);

        % code to store the last value of X,Y and Z for the second code
        if i<split_row
```

```
            X_1 = V(i,1);
            Y_1 = V(i,2);
            Z_1 = V(i,3);
        end

        if i<split_row

            if flag == 1
                fprintf(fid1,'G01 X%f Y%f Z%f E%f \n',X,Y,Z,E(i)+E2);
            else
                fprintf(fid1,'G01 X%f Y%f Z%f E%f \n',X,Y,Z,E(i)+E1);
            end
        else
            if flag == 1
                fprintf(fid2,'G01 X%f Y%f Z%f E%f \n',X,Y,Z,E(i)+E2);
            else
                fprintf(fid2,'G01 X%f Y%f Z%f E%f \n',X,Y,Z,E(i)+E1);
            end
        end
    end


end

% end of file code
fprintf(fid1,'G01 X%f Y%f Z%f E0 \n',Xstart,Y_1,Z_1);
fprintf(fid1,'M84 \n');
fclose(fid1);
fprintf(fid2,'G01 X%f Y%f Z%f E0 \n',Xstart,Ystart,V(end-1,3));
fprintf(fid2,'M84 \n');
fclose(fid2);

end
```
Published with MATLAB® R2016a

Section IV: G-Code

Flower G-Code

```
M106 S200
T0
G90
G21
G92 X0 Y0 Z0 E0
G0 X0 Y0 Z0 E0
G01 X15.900000 Y60.000000 Z0.000000 E1.000000 F400
G01 X85.900000 Y60.000000 Z0.000000 E1.455027
G01 X85.900081 Y59.974213 Z0.000000 E1.455195
G01 X85.900324 Y59.948428 Z0.000000 E1.455363
G01 X85.900730 Y59.922644 Z0.000000 E1.455530
G01 X85.901297 Y59.896863 Z0.000000 E1.455698
G01 X85.902027 Y59.871087 Z0.000000 E1.455865
G01 X85.902919 Y59.845316 Z0.000000 E1.456033
G01 X85.903973 Y59.819550 Z0.000000 E1.456201
G01 X85.905189 Y59.793792 Z0.000000 E1.456368
G01 X85.906567 Y59.768042 Z0.000000 E1.456536
G01 X85.908107 Y59.742302 Z0.000000 E1.456704
G01 X85.909808 Y59.716571 Z0.000000 E1.456871
G01 X85.911672 Y59.690852 Z0.000000 E1.457039
G01 X85.913697 Y59.665144 Z0.000000 E1.457206
G01 X85.915884 Y59.639450 Z0.000000 E1.457374
G01 X85.918232 Y59.613771 Z0.000000 E1.457542
G01 X85.920742 Y59.588106 Z0.000000 E1.457709
G01 X85.923414 Y59.562458 Z0.000000 E1.457877
G01 X85.926246 Y59.536828 Z0.000000 E1.458045
G01 X85.929240 Y59.511215 Z0.000000 E1.458212
G01 X85.932394 Y59.485622 Z0.000000 E1.458380
G01 X85.935710 Y59.460049 Z0.000000 E1.458547
G01 X85.939186 Y59.434498 Z0.000000 E1.458715
G01 X85.942823 Y59.408969 Z0.000000 E1.458883
G01 X85.946621 Y59.383463 Z0.000000 E1.459050
G01 X85.950579 Y59.357982 Z0.000000 E1.459218
G01 X85.954697 Y59.332526 Z0.000000 E1.459386
G01 X85.958975 Y59.307097 Z0.000000 E1.459553
G01 X85.963413 Y59.281695 Z0.000000 E1.459721
G01 X85.968010 Y59.256321 Z0.000000 E1.459888
G01 X85.972767 Y59.230977 Z0.000000 E1.460056
G01 X85.977684 Y59.205663 Z0.000000 E1.460224
G01 X85.982759 Y59.180381 Z0.000000 E1.460391
G01 X85.987994 Y59.155131 Z0.000000 E1.460559
G01 X85.993387 Y59.129914 Z0.000000 E1.460727
G01 X85.998938 Y59.104732 Z0.000000 E1.460894
G01 X86.004648 Y59.079585 Z0.000000 E1.461062
G01 X86.010516 Y59.054475 Z0.000000 E1.461229
G01 X86.016542 Y59.029402 Z0.000000 E1.461397
G01 X86.022725 Y59.004368 Z0.000000 E1.461565
G01 X86.029066 Y58.979372 Z0.000000 E1.461732
G01 X86.035563 Y58.954418 Z0.000000 E1.461900
G01 X86.042218 Y58.929504 Z0.000000 E1.462068
G01 X86.049029 Y58.904633 Z0.000000 E1.462235
G01 X86.055996 Y58.879806 Z0.000000 E1.462403
G01 X86.063120 Y58.855022 Z0.000000 E1.462570
G01 X86.070399 Y58.830284 Z0.000000 E1.462738
G01 X86.077833 Y58.805592 Z0.000000 E1.462906
G01 X86.085423 Y58.780948 Z0.000000 E1.463073
G01 X86.093168 Y58.756351 Z0.000000 E1.463241
G01 X86.101067 Y58.731804 Z0.000000 E1.463409
G01 X86.109120 Y58.707307 Z0.000000 E1.463576
G01 X86.117328 Y58.682861 Z0.000000 E1.463744
```

Leaf G-Code

```
M106 S200
T0
G90
G21
G92 X0 Y0 Z0 E0
G0 X0 Y0 Z0 E0
G01 X-7.864000 Y60.000000 Z0.000000 E1.000000 F400
G01 X42.136000 Y60.000000 Z0.000000 E1.325020
G01 X42.246686 Y59.888340 Z0.000000 E1.326042
G01 X42.354686 Y59.778029 Z0.000000 E1.327045
G01 X42.461821 Y59.669043 Z0.000000 E1.328038
G01 X42.570865 Y59.561427 Z0.000000 E1.329034
G01 X42.685284 Y59.455318 Z0.000000 E1.330049
G01 X42.808904 Y59.350954 Z0.000000 E1.331100
G01 X42.945570 Y59.248671 Z0.000000 E1.332210
G01 X43.098808 Y59.148885 Z0.000000 E1.333399
G01 X43.271540 Y59.052068 Z0.000000 E1.334686
G01 X43.465848 Y58.958715 Z0.000000 E1.336087
G01 X43.682837 Y58.869305 Z0.000000 E1.337613
G01 X43.922584 Y58.784269 Z0.000000 E1.339266
G01 X44.184184 Y58.703959 Z0.000000 E1.341045
G01 X44.465873 Y58.628633 Z0.000000 E1.342941
G01 X44.765217 Y58.558439 Z0.000000 E1.344939
G01 X45.079351 Y58.493422 Z0.000000 E1.347024
G01 X45.405223 Y58.433537 Z0.000000 E1.349178
G01 X45.739832 Y58.378666 Z0.000000 E1.351382
G01 X46.080439 Y58.328649 Z0.000000 E1.353620
G01 X46.424716 Y58.283308 Z0.000000 E1.355877
G01 X46.770840 Y58.242474 Z0.000000 E1.358143
G01 X47.117515 Y58.206007 Z0.000000 E1.360409
G01 X47.463935 Y58.173803 Z0.000000 E1.362671
G01 X47.809690 Y58.145800 Z0.000000 E1.364925
G01 X48.154631 Y58.121969 Z0.000000 E1.367173
G01 X48.498725 Y58.102293 Z0.000000 E1.369413
G01 X48.841899 Y58.086751 Z0.000000 E1.371647
G01 X49.183909 Y58.075291 Z0.000000 E1.373871
G01 X49.524241 Y58.067802 Z0.000000 E1.376084
G01 X49.862053 Y58.064101 Z0.000000 E1.378280
G01 X50.196172 Y58.063916 Z0.000000 E1.380452
G01 X50.525133 Y58.066886 Z0.000000 E1.382590
G01 X50.847260 Y58.072568 Z0.000000 E1.384684
G01 X51.160783 Y58.080454 Z0.000000 E1.386723
G01 X51.463971 Y58.089998 Z0.000000 E1.388695
G01 X51.755259 Y58.100648 Z0.000000 E1.390590
G01 X52.033380 Y58.111876 Z0.000000 E1.392399
G01 X52.297456 Y58.123220 Z0.000000 E1.394117
G01 X52.547063 Y58.134304 Z0.000000 E1.395741
G01 X52.782260 Y58.144864 Z0.000000 E1.397272
G01 X53.003565 Y58.154757 Z0.000000 E1.398712
G01 X53.211909 Y58.163966 Z0.000000 E1.400067
G01 X53.408547 Y58.172586 Z0.000000 E1.401347
G01 X53.594950 Y58.180808 Z0.000000 E1.402560
G01 X53.772686 Y58.188887 Z0.000000 E1.403716
G01 X53.943304 Y58.197116 Z0.000000 E1.404827
G01 X54.108216 Y58.205790 Z0.000000 E1.405900
G01 X54.268615 Y58.215169 Z0.000000 E1.406945
G01 X54.425396 Y58.225455 Z0.000000 E1.407966
G01 X54.579126 Y58.236766 Z0.000000 E1.408968
G01 X54.730026 Y58.249121 Z0.000000 E1.409952
G01 X54.877987 Y58.262433 Z0.000000 E1.410918
G01 X55.022610 Y58.276512 Z0.000000 E1.411862
G01 X55.163262 Y58.291072 Z0.000000 E1.412782
G01 X55.299141 Y58.305749 Z0.000000 E1.413670
G01 X55.429349 Y58.320123 Z0.000000 E1.414521
G01 X55.552967 Y58.333735 Z0.000000 E1.415330
```

Section V: Path Simulated

The G-Code was opened with Repetier. Fig. 28, 29 and 30 shows the path simulated. All figures show the two separate G-code used for the final print in which syringe switching was utilized. In Fig. 28 the bottom image represents the first G-code file and the top represents the second. It can be seen that the second picks up where the first left off to complete the print. In Fig. 29 and 30 the right shows the first G-code file and the left shows the second G-code file.

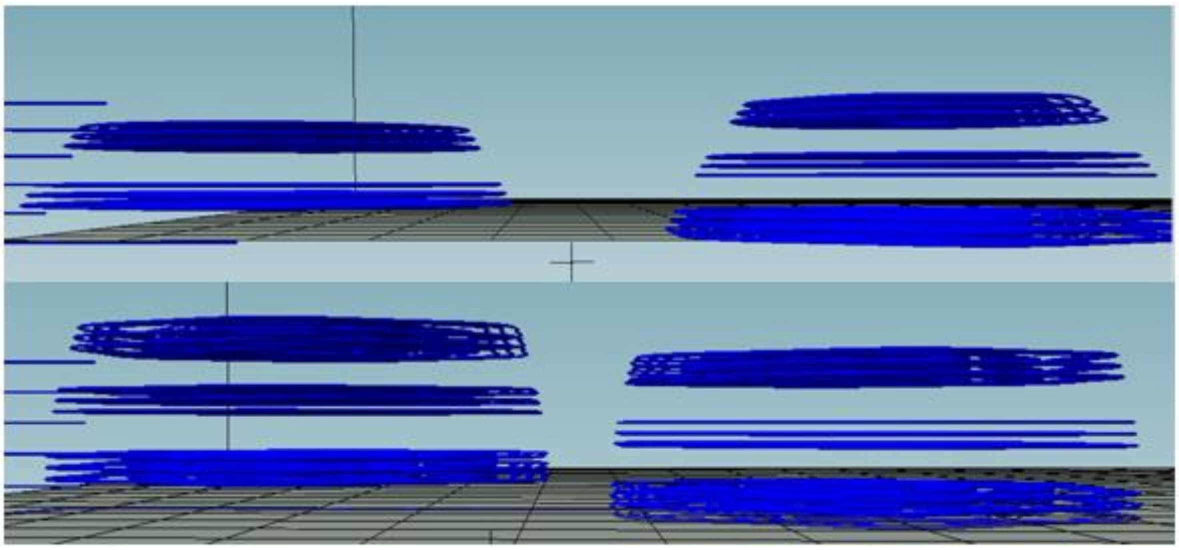
Figure 28: Side view of Simulated path, for both the top and bottom prints

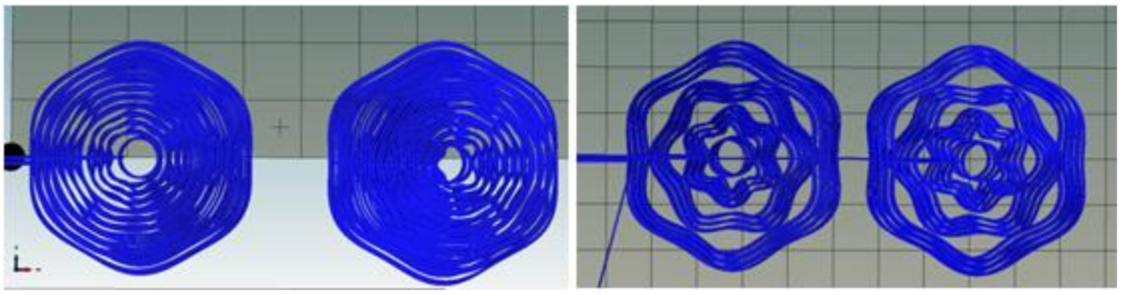
Figure 29: Simulated path - Top view

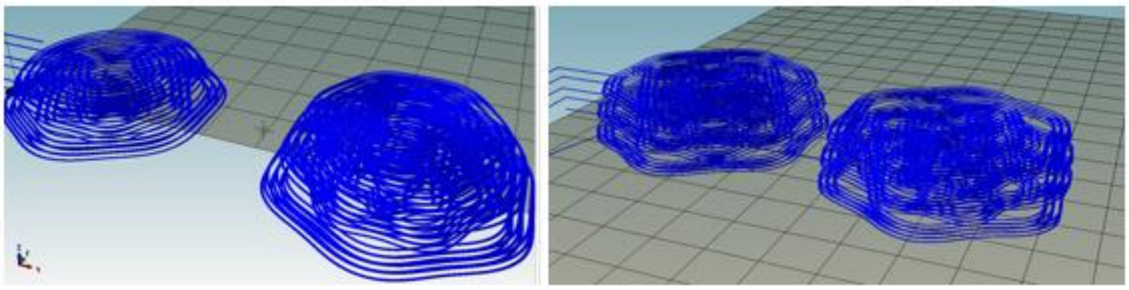
Figure 30: Simulated path - Isometric view

Section VI: Problems Encountered, Solution

Material Selection

New materials were tried in addition to Tofu and dough. The first material was a mixture of avocado and semolina flour, called Avoca-dough. The second material was a mixture of Nutella and flour, called Doughtella. The recipes for these two materials can be seen in Table.1 below and the materials can be seen in Fig. 35.

Table 1

Avoca-dough	**Doughtella**
½ an Avocado - Blend Well	½ Cup Nutella
⅛ Cup Semolina Flour (Bob's Red Mill)	⅛ Cup Semolina Flour (Bob's Red Mill)
⅛ Cup Sugar (Extra Fine White Sugar)	

The flour gave the main materials, avocado and Nutella, a viscous property that allowed it to hold form when printed in intricate designs. It also allowed for the materials to bake and harden without losing shape since it contained ingredients that do not expand under heat. Sugar was added to the Avoca-dough to enhance flavor, but also to show that granular ingredients could be added without negatively affecting the print results.

The avocado had to be blended to ensure no large chunks remained that could clog the extrusion tip. When printed, Avoca-dough had a smoother texture compared to To-dough. Baking the Avoca-dough produced a nice surface finish, and the material was able to hold its shape without expanding. It was determined that baking the print in a toaster oven is a better process compared to using the microwave. The toaster oven provided a more gradual and controlled heating of the material, which made it ideal for shape retention.

Several problems were encountered during material preparation and during the printing process. Similar to previous materials, air bubbles during the syringe filling process were difficult to avoid. Again, to solve this problem, a chopstick was dipped into canola oil and then rubbed along the inside wall of the syringe. This coat of oil helped the filling process by allowing the material to fall to the bottom of the syringe, which reduced pockets of air that would get trapped in between insertion layers. The oil also reduced friction while the material was extruding, which helped overcome chunks in the material that would impede the extrusion rate. Preparation of the Avoca-dough had to be performed with care. The inner skin of the avocado, along with its pit at the center, can drop contaminants (brown hard chunks) into the mixture. These contaminants would clog up the tip of the nozzle during the extrusion process, stopping the flow of material and leaving a gap in the print. This problem was encountered during the first few iterations.

Another issue encountered while working with Avoca-dough was the hardening of the mixture during excess exposure to air. When the Avocado-dough was left out in the open, its surface began to harden after 20 minutes. The hardened avocado surface had a brown, dry texture, and would clog up the nozzle tip during the printing process. This issue was avoided by meticulously picking out and removing the brown contaminants during the syringe filling process. Any excess mixture containing avocado was also put into a Ziploc bag to reduce the rate of oxidization. When a syringe that has been exposed in air for a prolonged period is used, it was necessary to scrape off the oxidized avocado at the top.

Baking with two different materials proved to be challenging as well. The Doughtella became soft when baked, and flowed into the crevices in the flower design. As the baking time increased, the Doughtella layers also began to collapse, and the overall structure started to sink in height. While the overall structure held its shape after the baking process, the inside pattern of the flower was distorted and filled with excess Doughtella.

Iteration 1

The first iteration of the print was made with Avoca-dough and designed for 20 layers. The foundation layer had a diameter of 30mm with three walls, and the result can be seen in Fig. 31 (pre-baking on the left). The print had the shape of a flower tapered parabolically towards the top. The texture of the printed design was very uniform, and the material stacked quite well. The finished print was put into the oven and baked at 250^{o} F for 10 minutes. The baked product can be seen in the right picture of Fig. 31. The baking, as well as the expansion, appeared to be uniform throughout the print. The shape of the product appeared to be shorter in height due to the transferring process from the parchment paper to the baking pan. Additionally, the parchment paper warped when in contact with the print and the area in contact with the paper became deformed. It was noted that future prints should be done directly onto the baking pan, simplifying the baking process and eliminating the deformities.

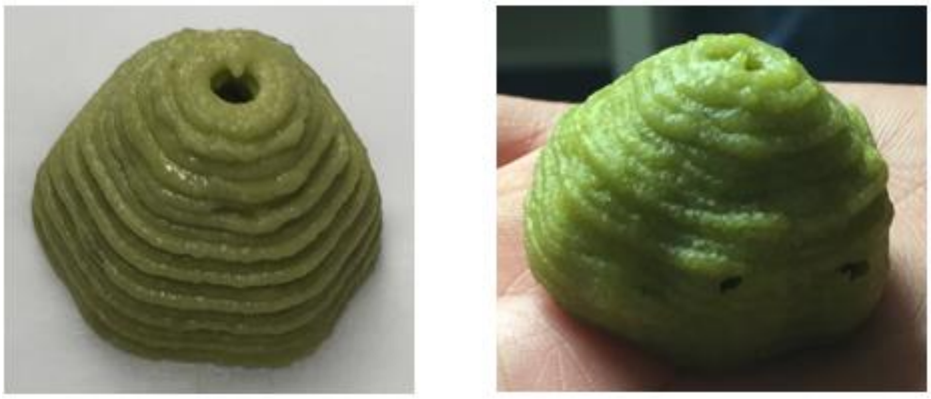

Figure 31: 15 loop iteration

Iteration 2

The print for iteration 2 can be seen in Fig.21. It comprises of a flower (center picture), made with two materials, and two maple leafs. For the flower, the first material is a mixture of avocado and semolina flour, called Avoca-dough; while the second material is a mixture of Nutella and semolina flour, called Doughtella. The maple leaf was printed solely with the Avoca-dough. Food coloring was used to create the red version.

A problem faced in this iteration was that the tips of the syringe were at different Z-heights, even if the surface was perfectly flat. This was because the tip of the syringe is manually screwed in, and it is difficult to get both tips to be tightened to the exact same amount. Thus, every time a new batch of syringes were loaded, the tips were aligned. An example of this can be seen in this video where a tip had to be replaced when a contaminant from the Avoca-dough pit clogged the nozzle. Once screwed in, the tightness of the fit is adjusted to get the Z-heights perfectly calibrated.

An issue that can been seen from the video was that the G-code contained negative values for the priming line right after the syringe switching occurred. These values were not recognized by the machine and so when the print resumed it was offset in the positive X direction by the magnitude of the negative value. Manual adjustment had to be made in order to continue with the print.

The Doughtella lost from when baked as explained previously; therefore, a third iteration was necessary.

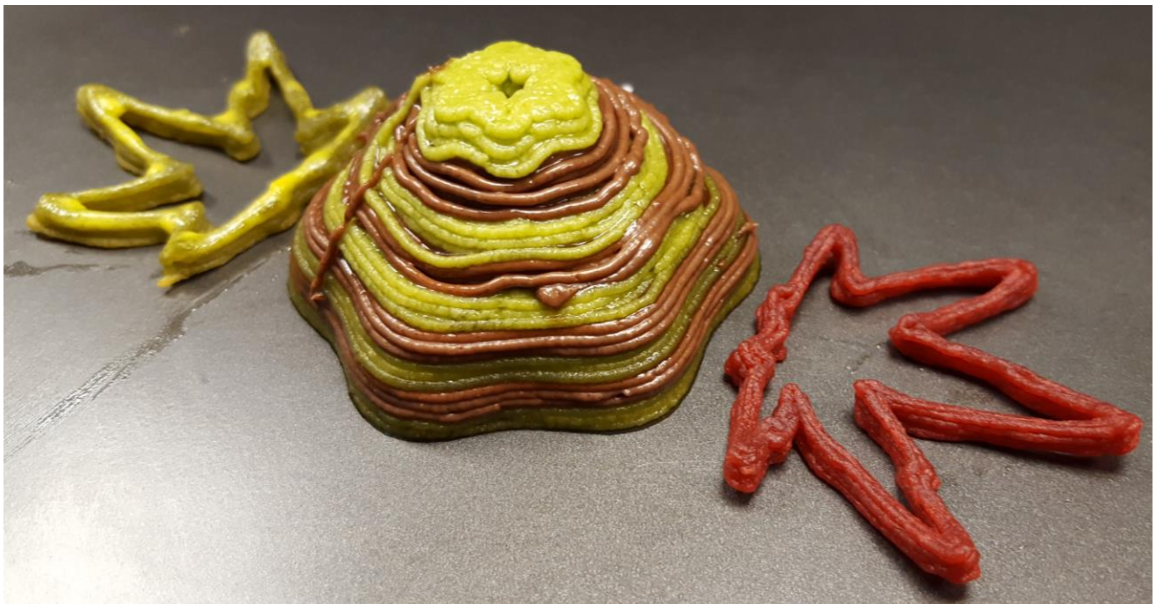

Figure 32: Flower (37 layers) with maple leaf (4 layers)

Iteration 3

For the final print (left image in Fig. 33, pre-baking) the two materials used were both Avoca-Dough, one with its natural green color and the other with red food coloring for aesthetic touch. The number of layers and the diameter of the bottom layer was decreased in order to improve shape retention. Also an important improvement made was decreasing the change in Z-height linearly by .0047 mm per layer as the height of the structure increased. At the start the change in Z-height between layers was 1.05 mm and by the end it was .95 mm. The negative values appearing in the priming line after the syringe switch were fixed and the intermediate code to lock the motors was implemented. Both allowed for a seamless transition and perfect stacking of layers. Additionally, the rate of decrease in layer radius was reduced in order to improve stability.

Upon completion the flower structure was baked at 250 degrees F for 20 minutes and the result is shown on the right of Fig. 33 and 34. Five leaves, all made of Avoca-dough, two green and three red are printed with two walls and four layers and are used to garnish the dish. The plating arrangement can be seen in Fig. 34.

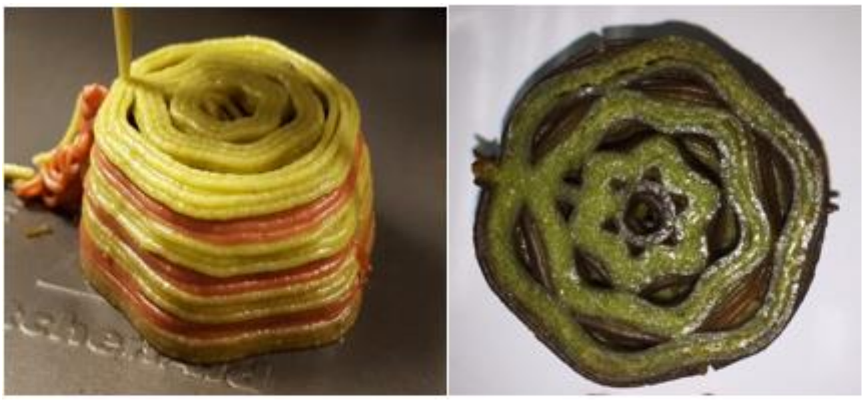

Figure 33: 33 layers (Left: pre-baking, Right: post-baking)

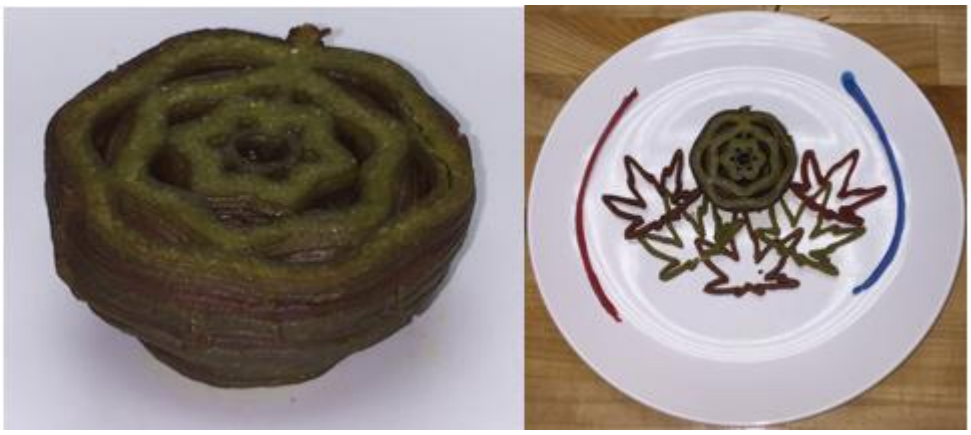

Figure 34: Baked Flower (left), Plating (right)

Figure 35: Materials used

Section VII: Videos

There were many videos shot of the process through the iterations and uploaded to YouTube.

A timelapse video of the material mixing process and loading into the syringe can be seen here: https://www.youtube.com/watch?v=dDnF0MVhAEY

A timelapse video of printing the leaf garnish can be seen here: https://www.youtube.com/watch?v=5zSRpZvSGvA

A timelapse video of a print in which the tip clogged and was replaced. The Z-height of the tips had to be re-calibrated and can be seen here: https://www.youtube.com/watch?v=PZd5FBDqckM

A timelapse video of iteration 2 can be seen here: https://www.youtube.com/watch?v=YfLxbyYGp84

A timelapse video of the final iteration can been seen here: https://www.youtube.com/watch?v=-rlYho5xKyA

A video of the finished product after baking can be seen here: https://www.youtube.com/watch?v=paffOCjH40E

Section VIII: Glamour Shots

Fig. 36-41 show some glamour shots of the final plating arrangement.

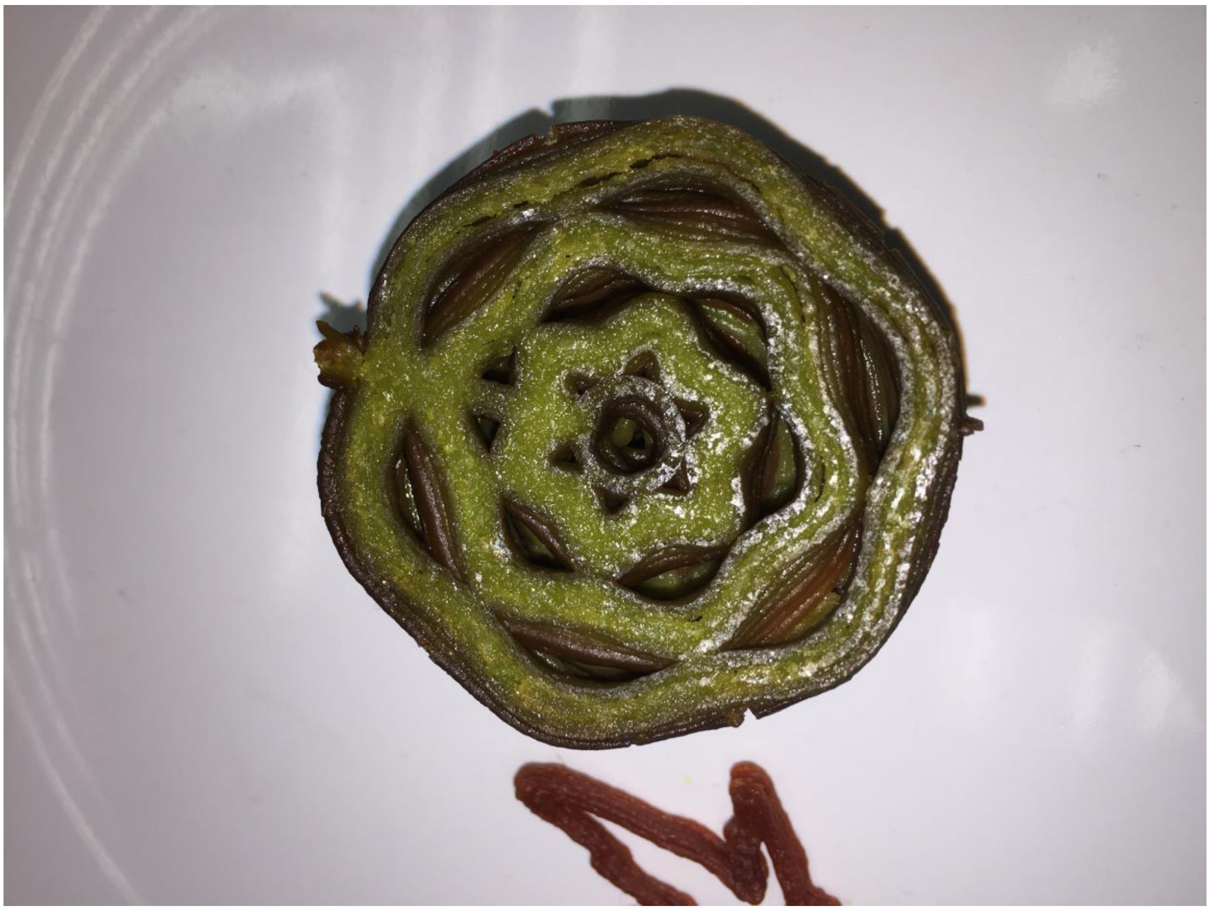

Figure 36: Glamour shot 1

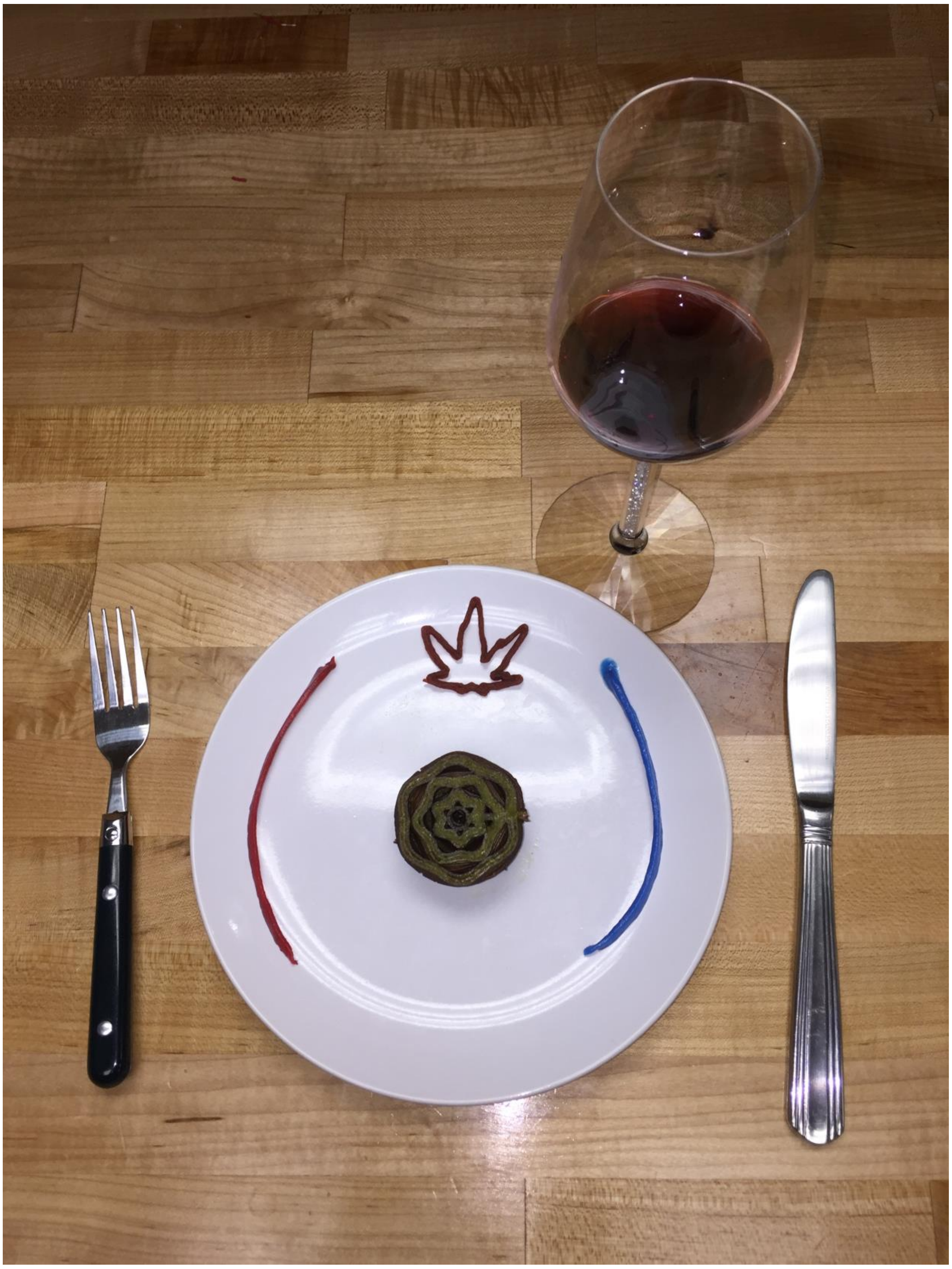

Figure 37: Glamour shot 2

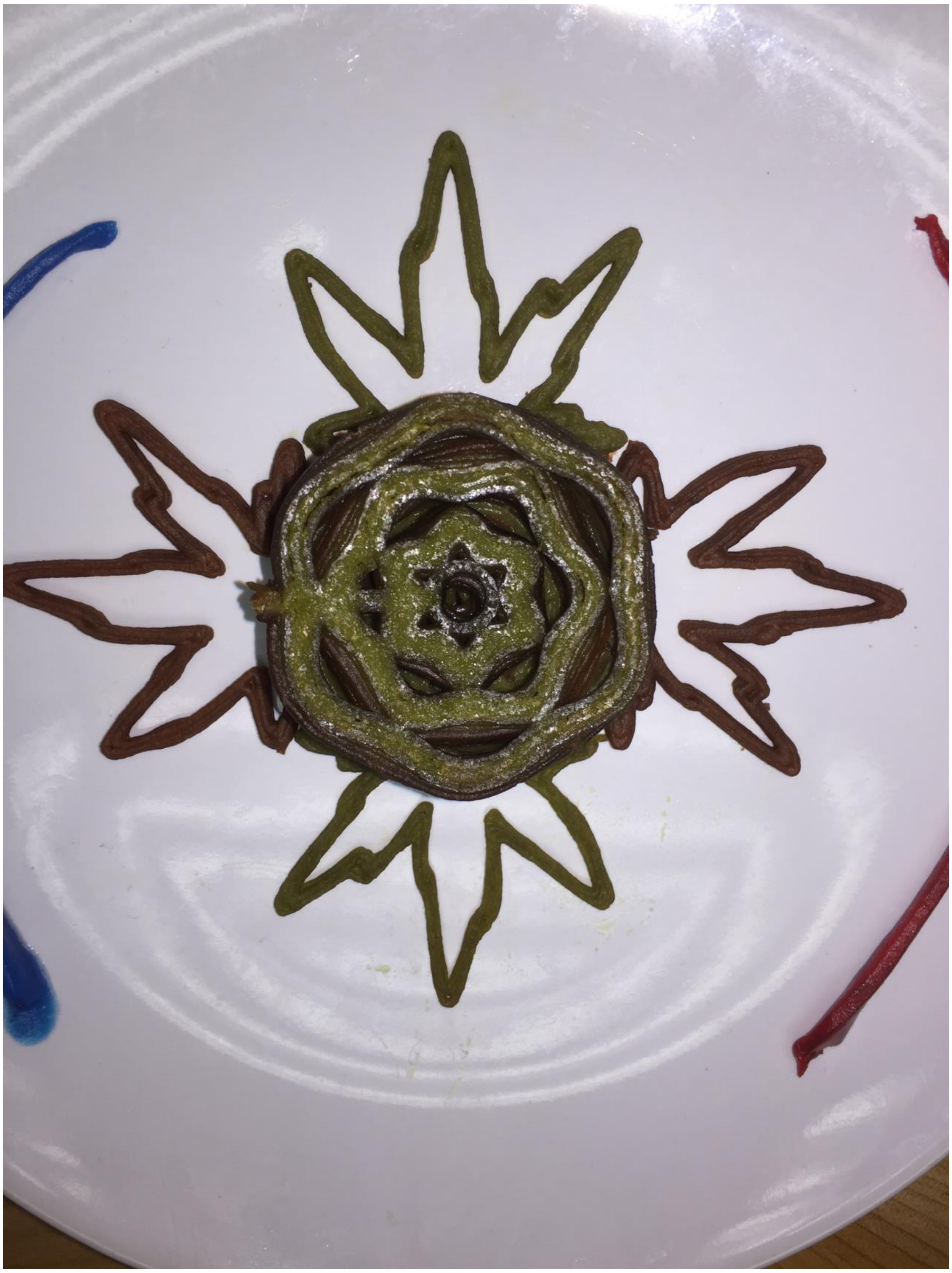
Figure 38: Glamour shot 3

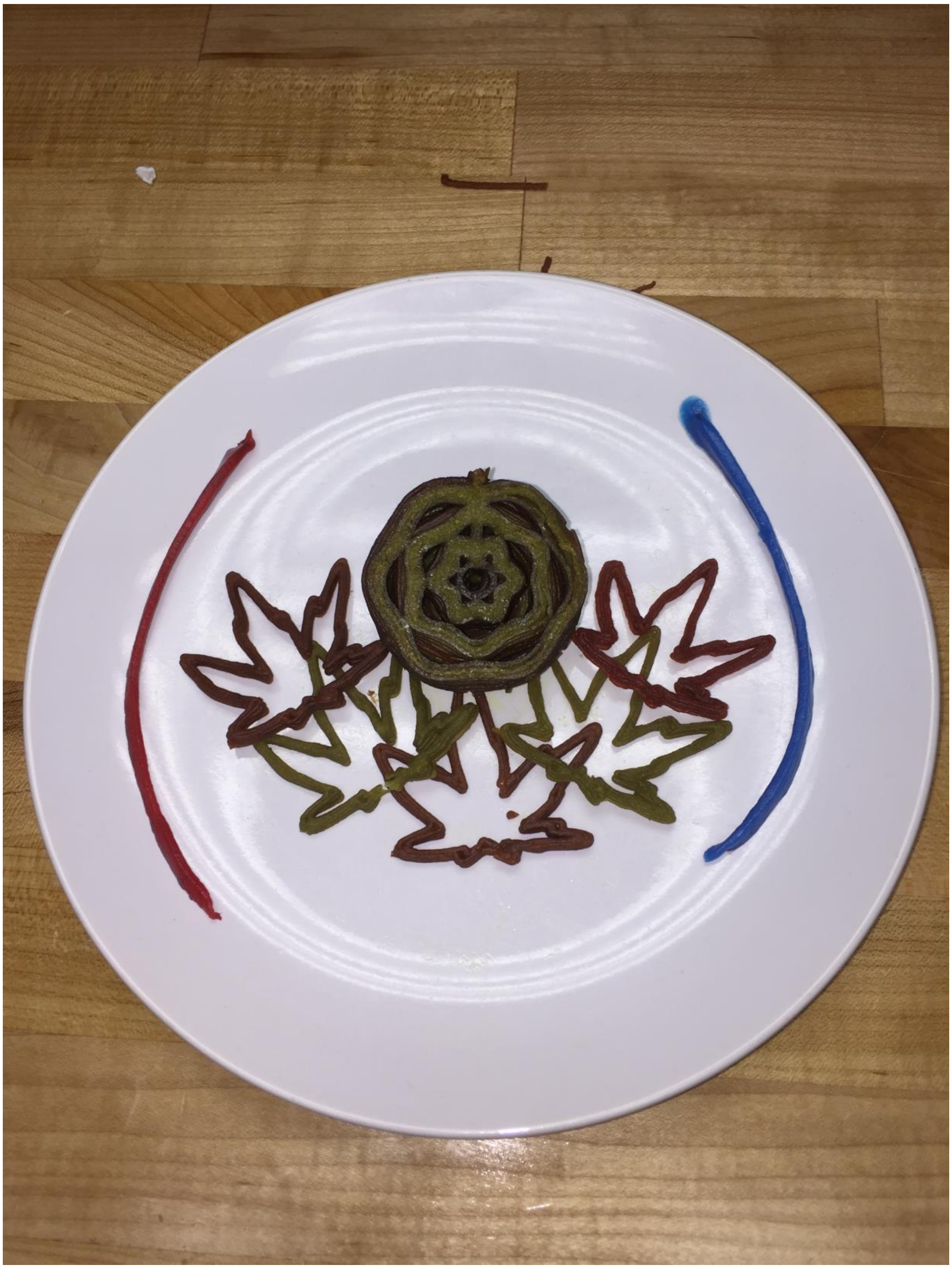

Figure 39: Glamour shot 4

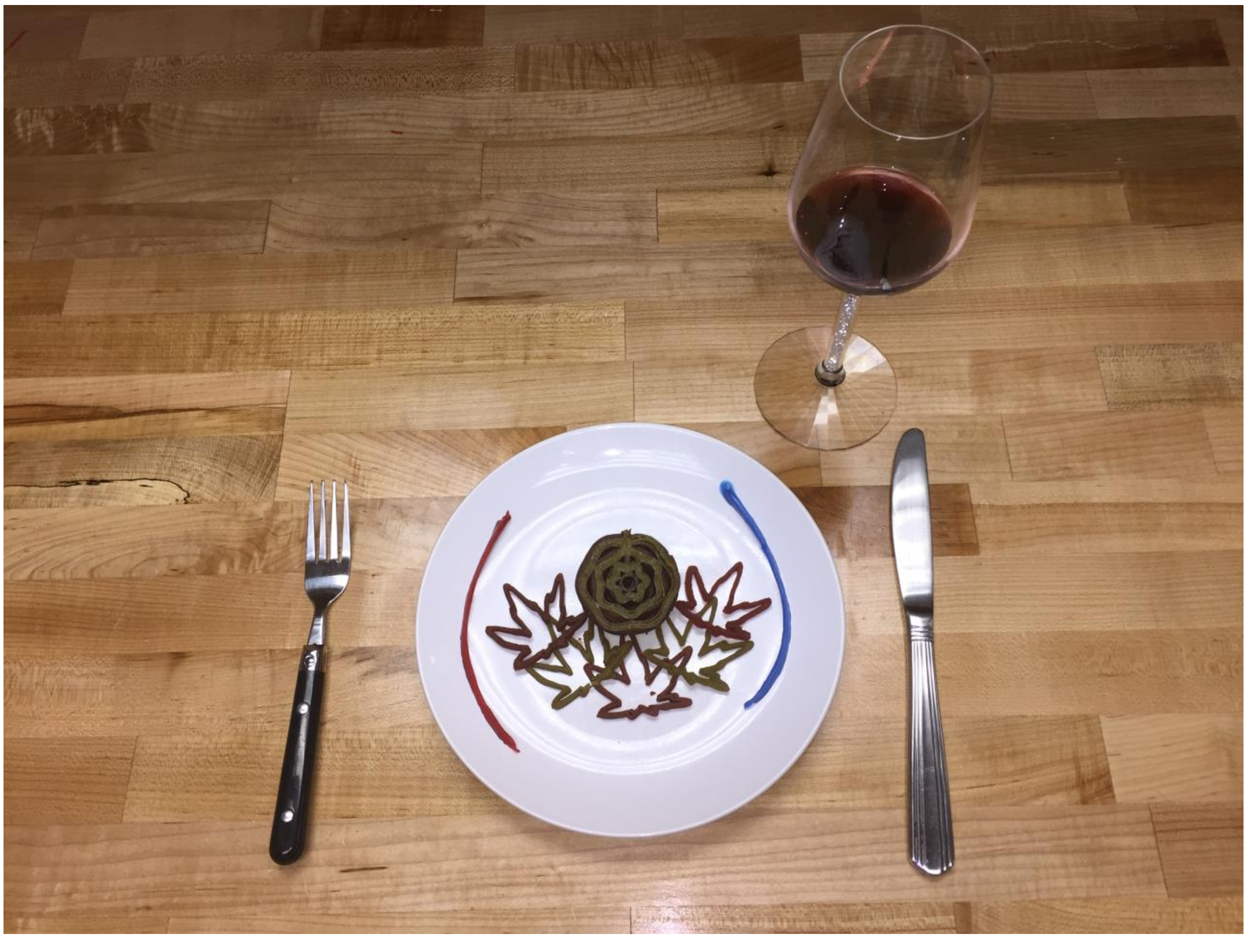

Figure 40: Glamour shot 5

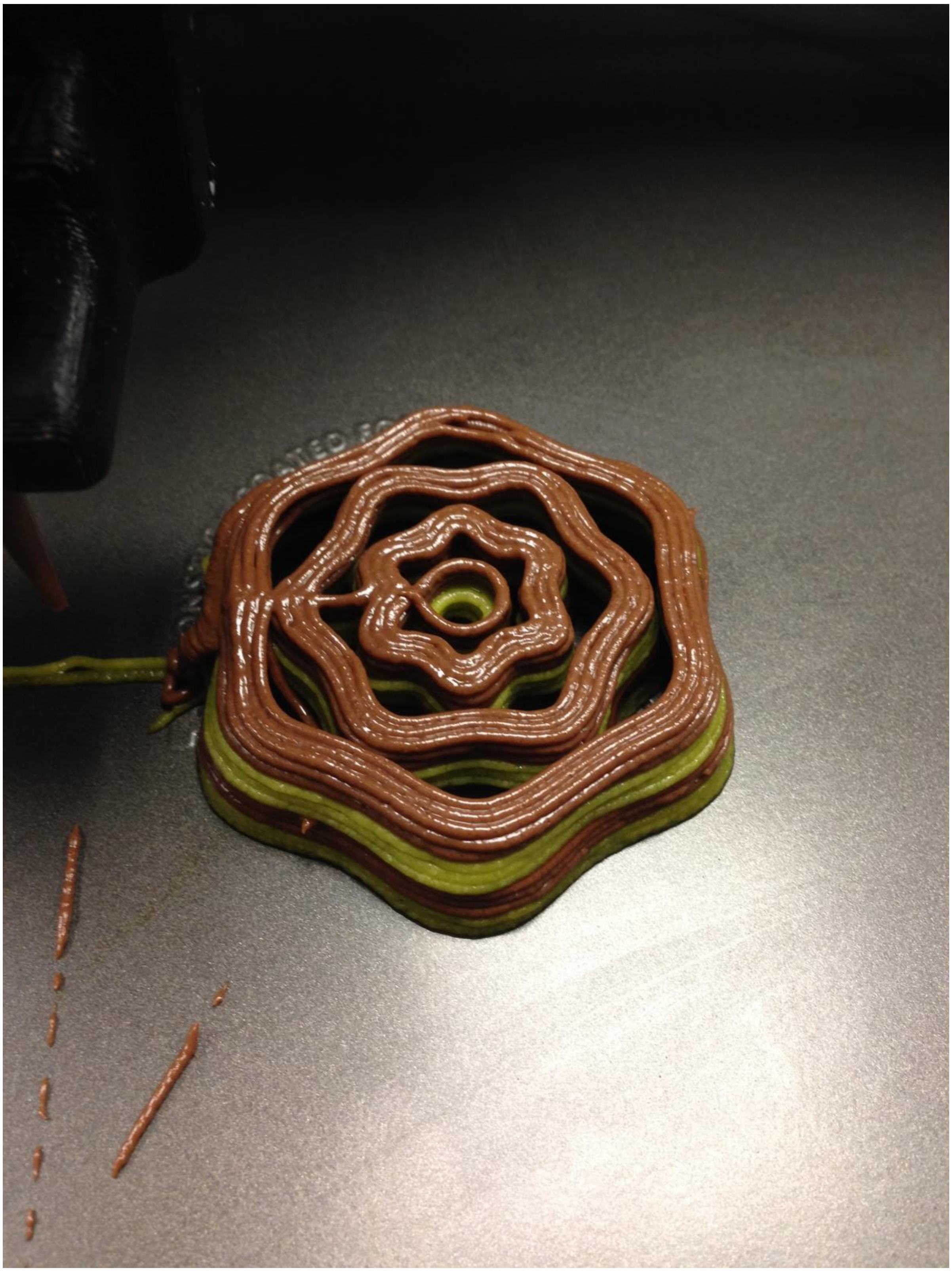

Figure 41: Glamour shot 6

Bibliography

[1] "G-Code Simulator". *Nraynaud.github.io*.

[2] "Food Printing - Group 7 Part 1". *drive.google.com/open?id=0B2MUkqzbA4s6SkVGczhoNnJwSFE*

[3] "Repetier Software". *repetier.com*

[4] "Hypocycloid Equation". *http://www-groups.dcs.st-and.ac.uk/history/Curves/Epicycloid.html*

[5] "Pod Collective: Polar Graph Art". *https://podcollective.com/polar-graph-art-quickgraph-a/*

[6] "The Cannabis Equation". *https://math.stackexchange.com/questions/618786/cannabis-equation*