

Contents

- Monte-Carlo Modeling of Electron Transport
- 1. Code Comments
- 2. Discussion
- 3. Conclusion

```
function assign1
```

Monte-Carlo Modeling of Electron Transport

ELEC 4700 Assignment - 1 Braden Bale (101072763)

1. Code Comments

First, given variables are assigned values where kB is Boltzmann constant, m0 is the rest mass of an electron, mn is the effective mass of an electron, iterations contains the amount of iterations to un through the for loop, dt is the given timestep, time is an array which contains each time step, T is the assumed temperature in K, tmn is the mean time between collisions as given in seconds, w is the width of the plotting area in nm, h is the height of the plotting area in nm, and NE is the number of electrons to be simulated.

```
set(0, 'DefaultFigureWindowStyle', 'docked')

close all

kB = 1.38066e-23;
m0 = 9.11e-31;
mn = 0.26*m0;

iterations = 75;

dt = 0.01e-12;

time = zeros(1, 1);

T = 300;
tmn = 0.2e-12;

w = 200e-9;
h = 100e-9;

NE = 10000;
```

From the setup variables above, the thermal velocity can be found as shown below.

```
thm_v = sqrt((2*kB*T)/mn);
```

When the thermal velocity is found the mean free path is found below as the thermal velocity multiplied by the mean time between collisions.

```
lmn = thm_v * tmn;
```

Finally, with the variables set up the initial electron positions are set using the randn() function.

```
x = rand(NE, 1)*w;
y = rand(NE, 1)*h;
```

To ensure electrons are not initialized in the boxes they are pushed to the left or right depending on their position.

```
for j = 1:NE
    if(x(j) < 120e-9) && (x(j) > 80e-9)
        if x(j) > 100e-9
            x(j) = x(j) + (20e-9);
        else
            x(j) = x(j) - (20e-9);
        end
    end
end
```

The previous x and y positions are initialized with the positions of the electrons.

```
xp = x;
yp = y;
```

The velocities of the electrons in the x and y direction are initialized using randn() and multiplying by half of the thermal velocity.

```
vx = 1*sqrt((kB*T)/mn)*randn(NE, 1);
vy = 1*sqrt((kB*T)/mn)*randn(NE, 1);
```

The boxes are initialized below by using classes.

```
box = {};
box{1}.x = [80e-9 120e-9];
box{1}.y = [60e-9];

box{2}.x = [80e-9 120e-9];
box{2}.y = [40e-9];
```

Figure 1 is a histogram representing the average velocity of each initialized electron.

```
figure(1)
histogram(sqrt(vx.^2 +vy.^2), 100);
title('Average Velocity Histogram')

ntraj = 10;
electrons = [1:ntraj];
TrajColours = hsv(ntraj);
```

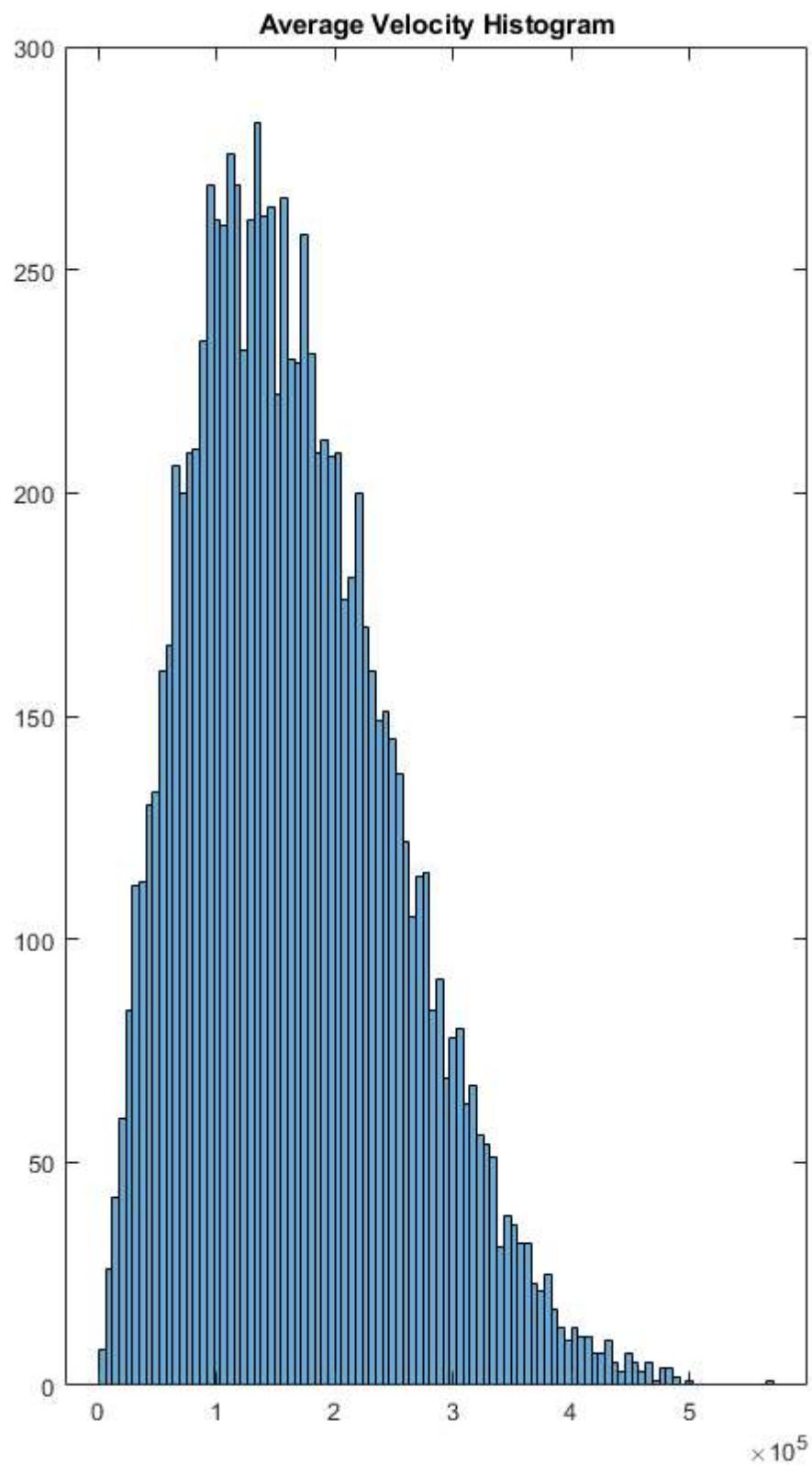
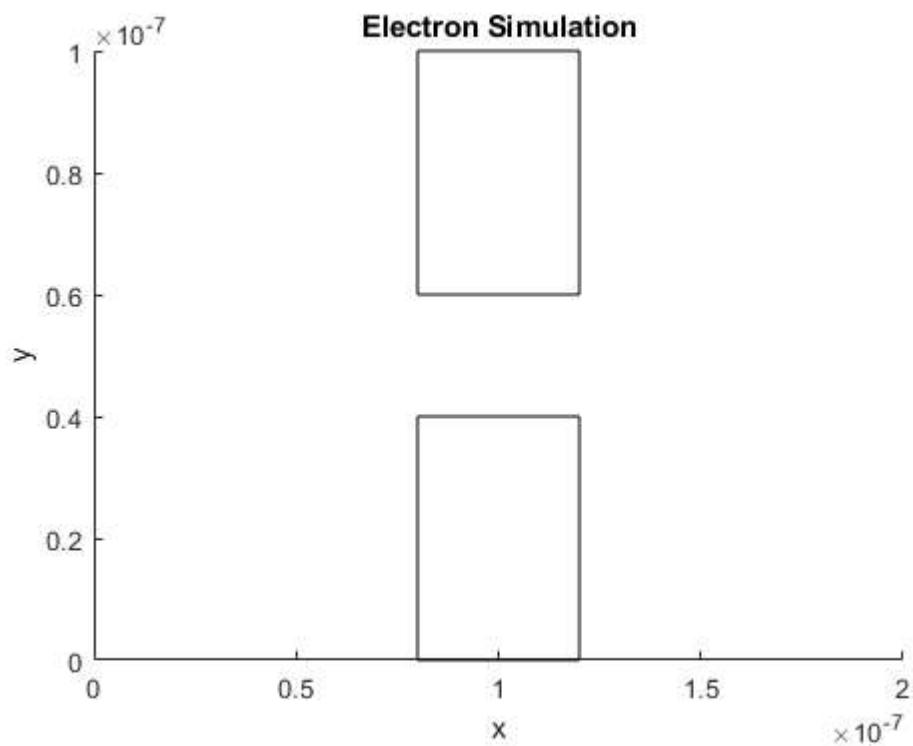


Figure 2 is set up using the `traj()` function which displays the electrons and their trajectory.

```
figure(2)
subplot(2,1,1)
hold on
traj(electrons, xp, x, yp, y, TrajColours, w, h);
pause(0.01);
axis([0 w 0 h])
```



A second part of Figure 2 is a plot of the average temperature over time for the electron plot.

```

avev = sqrt(vy.^2 + vx.^2);
avet = ((avev.^2)*mn)/(2*kB);
sumavt(1) = (sum(avet)/NE);

```

The for loop runs based off number of iterations.

```
for i = 1: iterations
```

First, a new time is added to the time array and the new x and y positions are updated.

```

time1 = i*dt;
time = cat(1, time, time1);
dx = vx*dt;
dy = vy*dt;
x = xp + dx;
y = yp + dy;

```

Lines 138 through 169 and 191 through 203 house limits to meet the expectations given. Where variables b1* and b2* deal with box one and two, y*t and x*t apply to the axes. Also, the commented code below changes the boxes to specular behaviour, while the currently active code is diffusive behaviour.

```

b1l = (x >= 80e-9) & (x <= 120e-9) & (xp <= 80e-9) & (y >= 60e-9);
b1r = (x >= 80e-9) & (x <= 120e-9) & (xp >= 120e-9) & (y >= 60e-9);
b1b = (x >= 80e-9) & (x <= 120e-9) & (yp <= 60e-9) & (y >= 60e-9);

b2l = (x >= 80e-9) & (x <= 120e-9) & (xp <= 80e-9) & (y <= 40e-9);
b2r = (x >= 80e-9) & (x <= 120e-9) & (xp >= 120e-9) & (y <= 40e-9);
b2t = (x >= 80e-9) & (x <= 120e-9) & (yp >= 40e-9) & (y <= 40e-9);

x(b1l) = box{1}.x(1);
x(b1r) = box{1}.x(2);
y(b1b) = box{1}.y;

x(b2l) = box{2}.x(1);
x(b2r) = box{2}.x(2);
y(b2t) = box{2}.y;

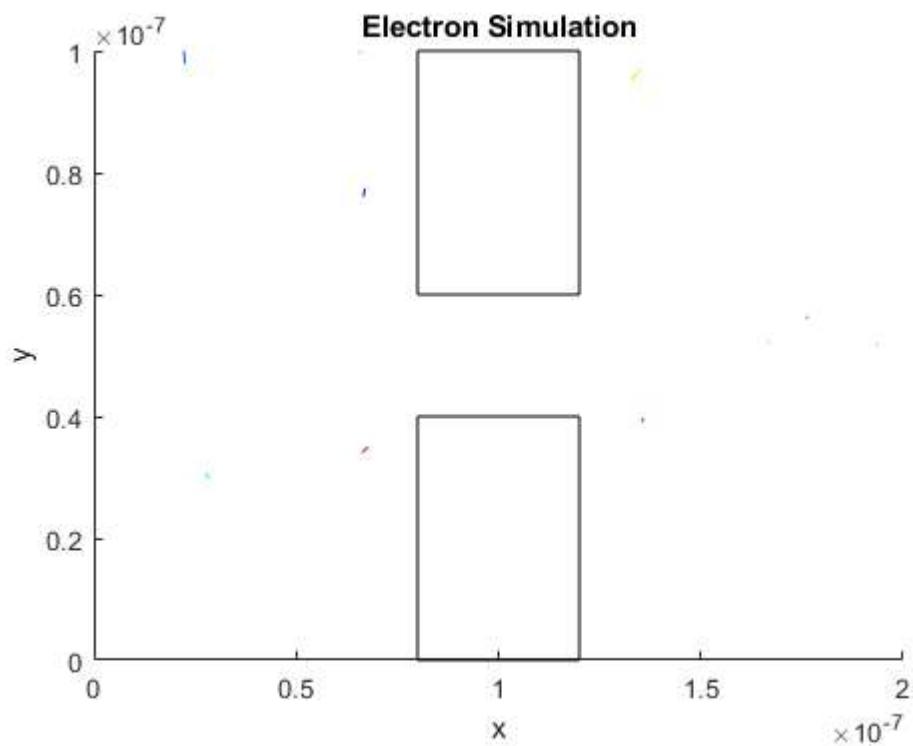
% SPECULAR
% vx(b1l) = -vx(b1l);
% vx(b1r) = -vx(b1r);
% vy(b1b) = -vy(b1b);
%
% vx(b2l) = -vx(b2l);
% vx(b2r) = -vx(b2r);
% vy(b2t) = -vy(b2t);
%
% DIFFUSIVE
vx(b1l) = -abs(1*sqrt((kB*T)/mn))*randn(length(NE), 1);
vx(b1r) = abs(1*sqrt((kB*T)/mn))*(randn(length(NE), 1));
vy(b1b) = -abs(1*sqrt((kB*T)/mn))*(randn(length(NE), 1));

vx(b2l) = -abs(1*sqrt((kB*T)/mn))*(randn(length(NE), 1));
vx(b2r) = abs(1*sqrt((kB*T)/mn))*(randn(length(NE), 1));
vy(b2t) = abs(1*sqrt((kB*T)/mn))*(randn(length(NE), 1));

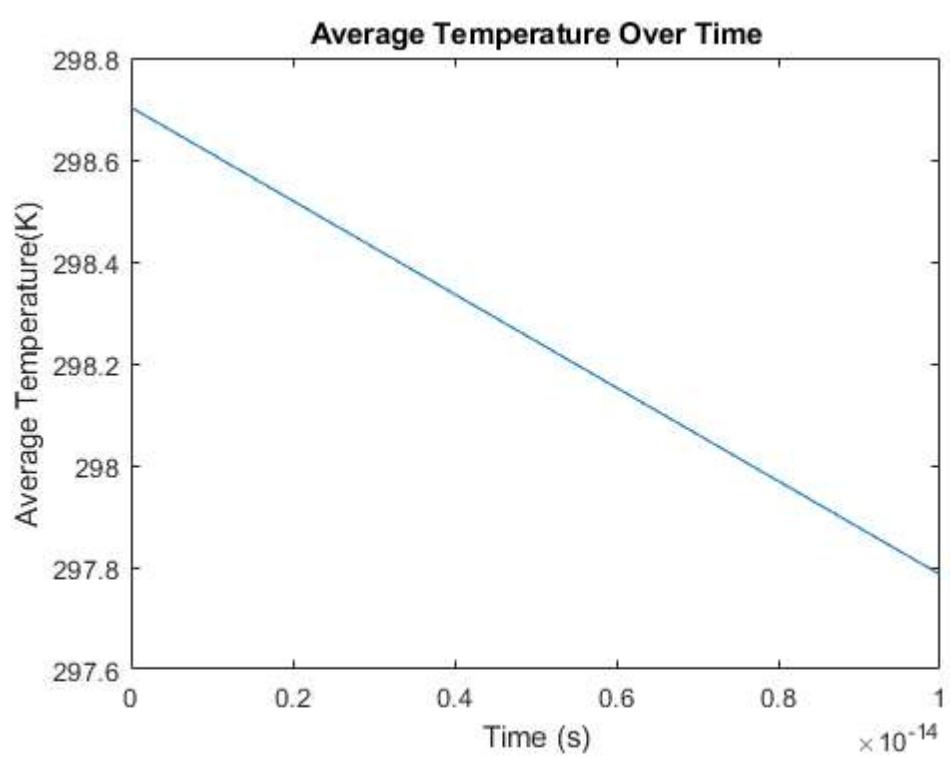
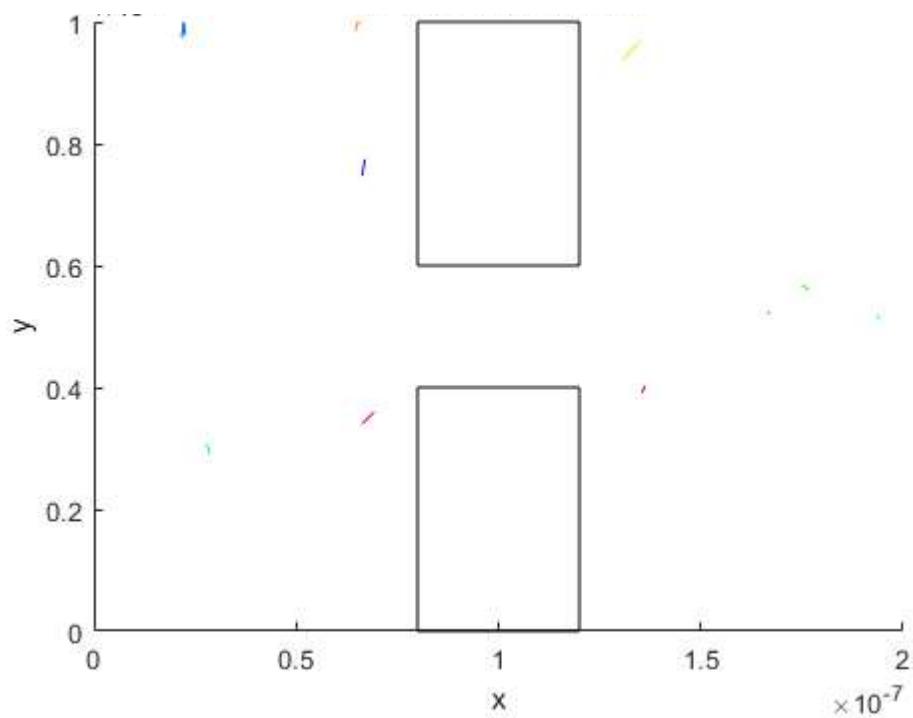
```

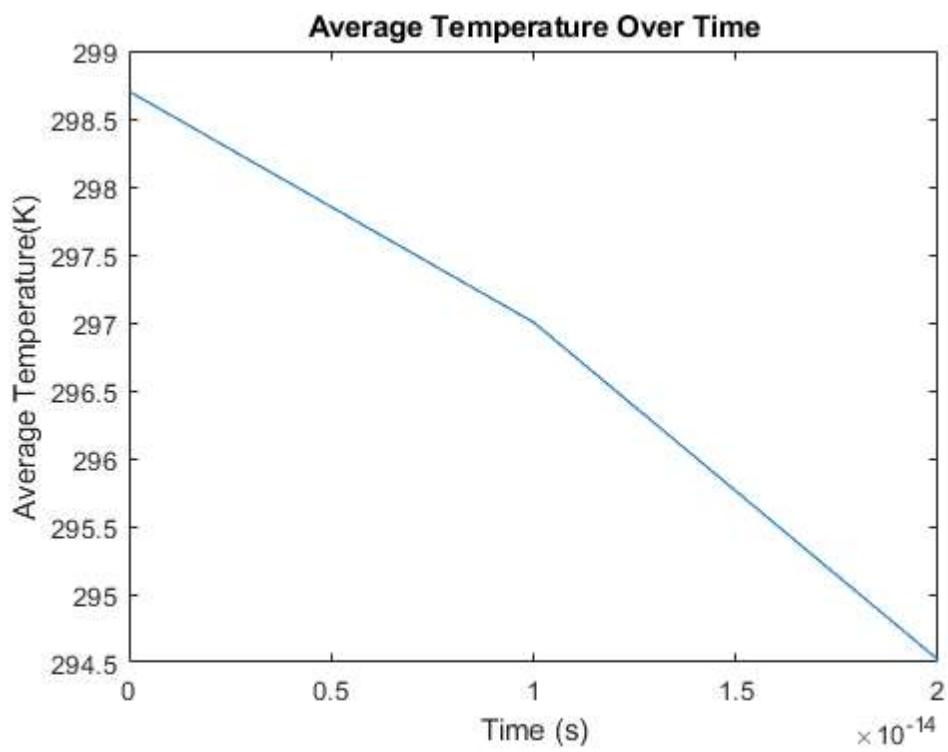
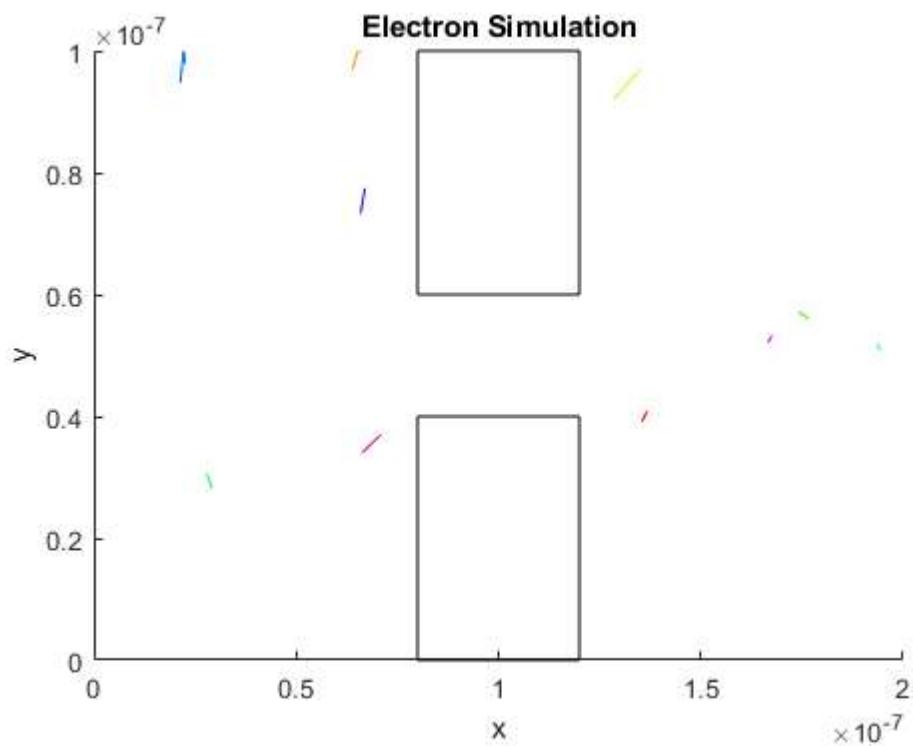
After having the limit tests applied the new positions of x and y along with their trajectories are plotted.

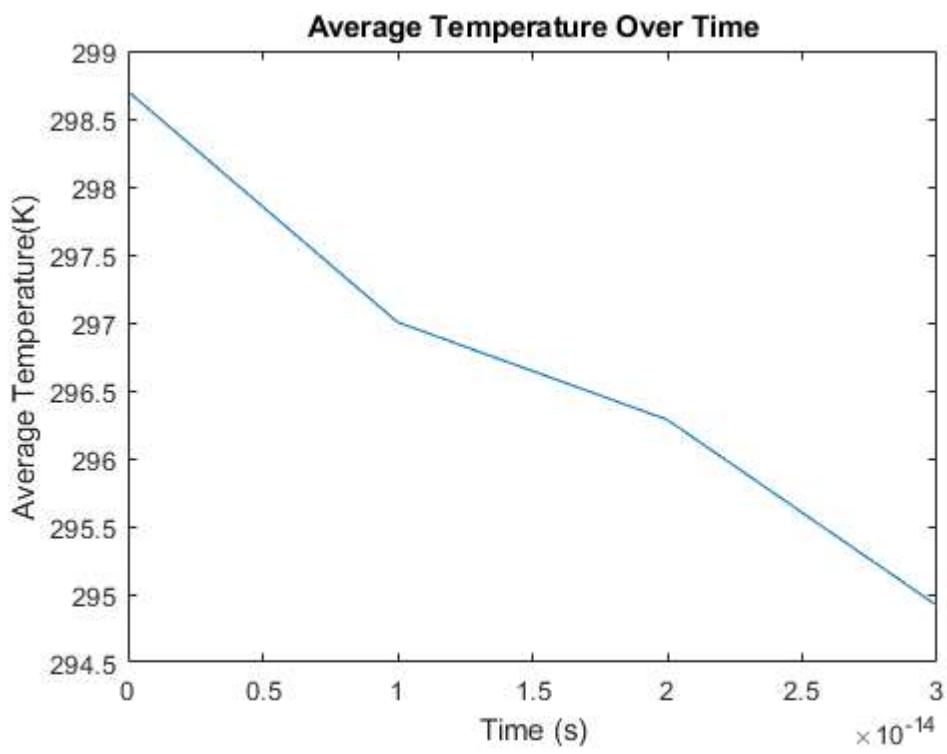
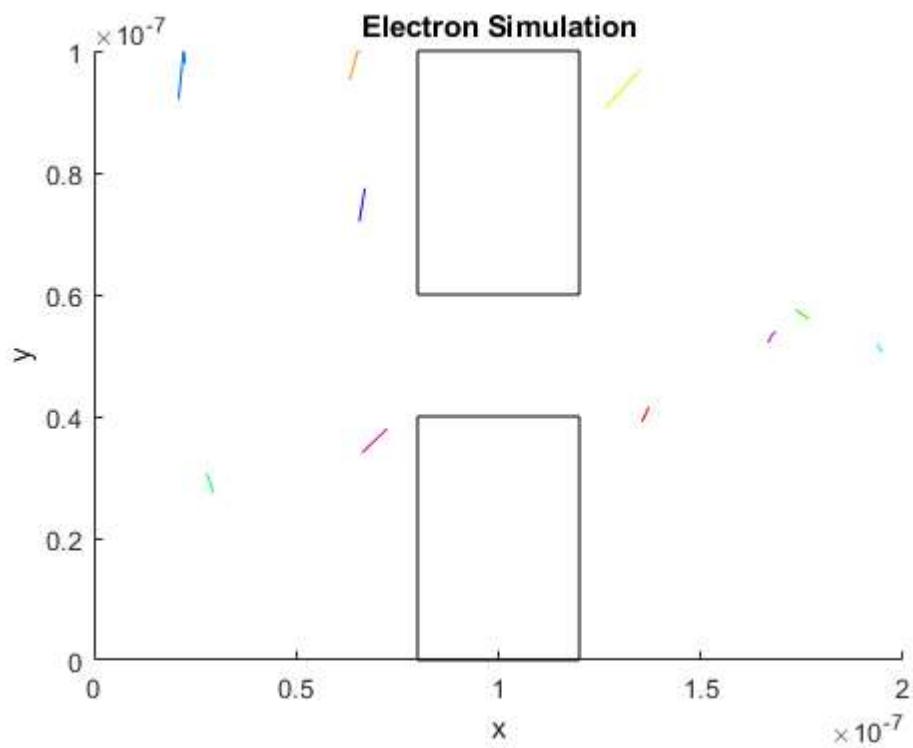
```
subplot(2,1,1)
traj(electrons, xp, x, yp, y, TrajColours, w, h);
pause(0.01);
```

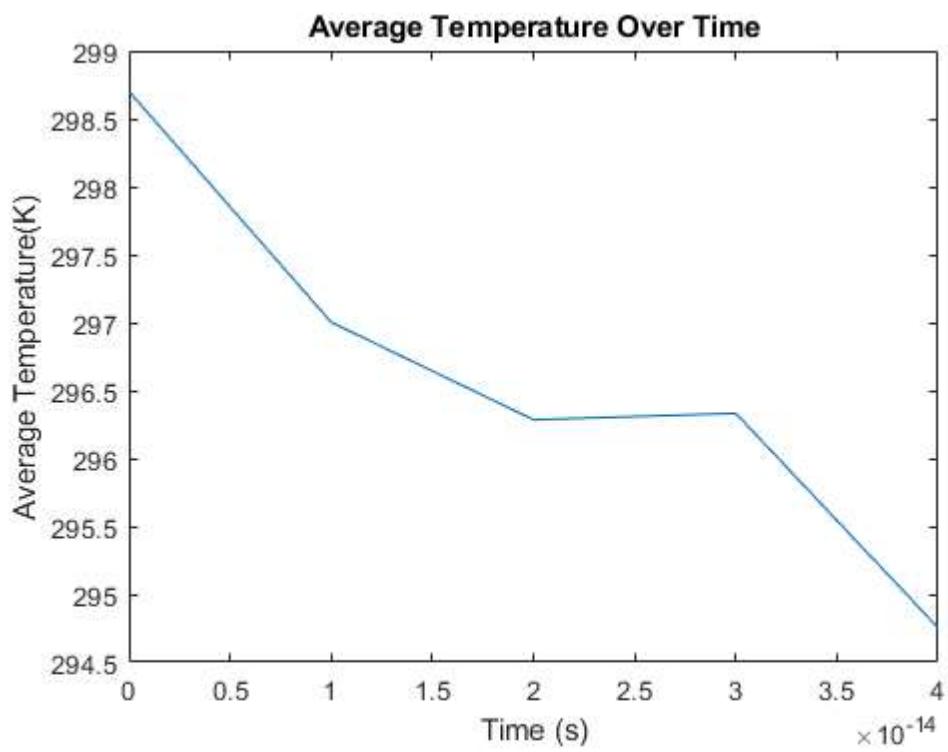
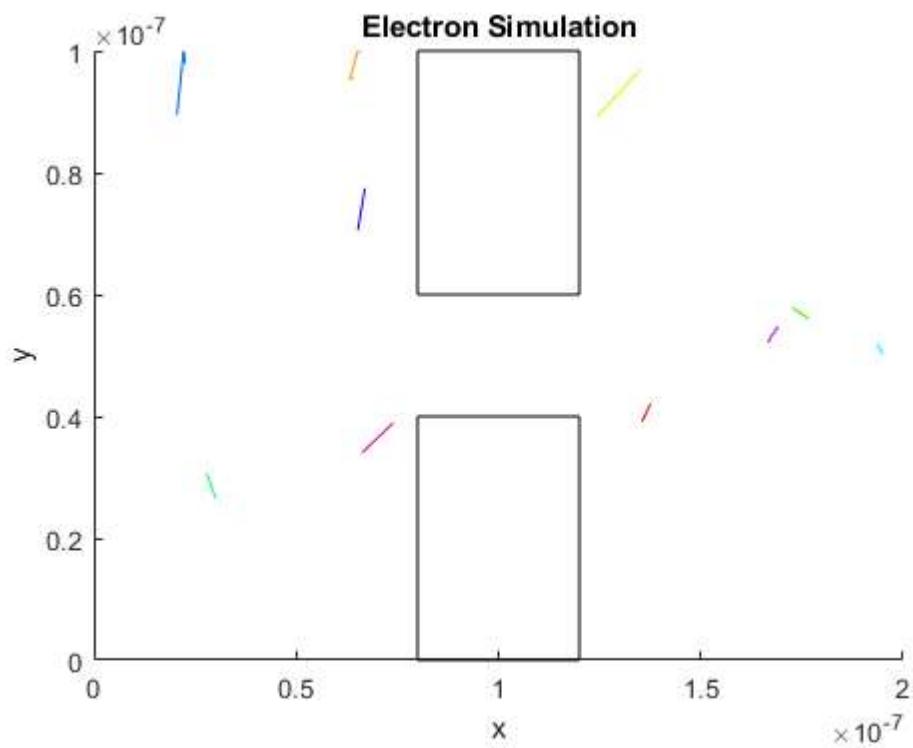


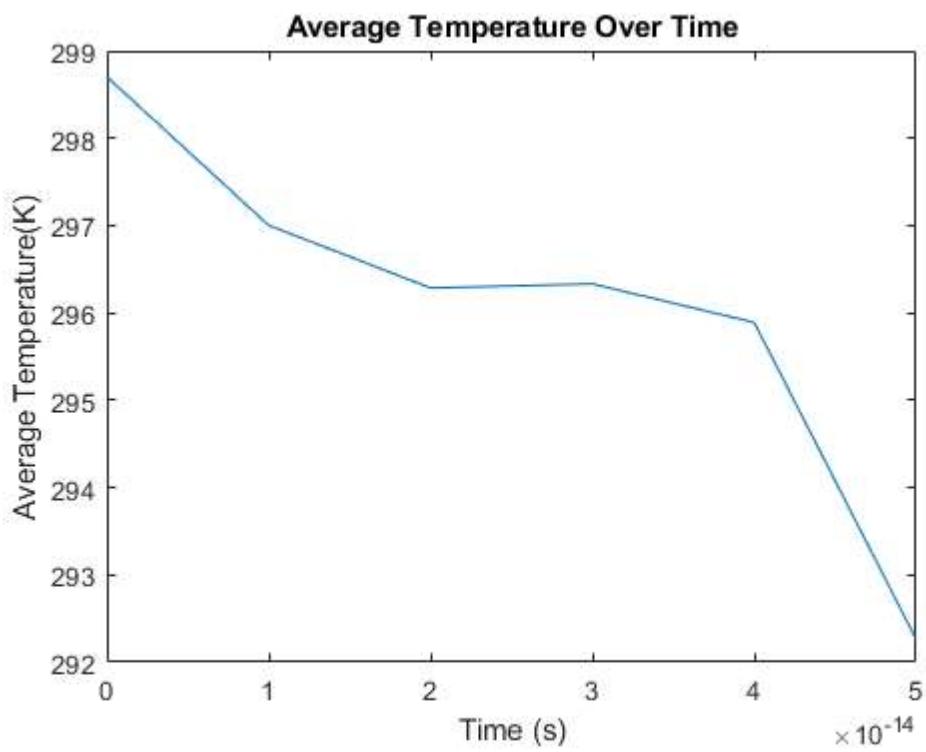
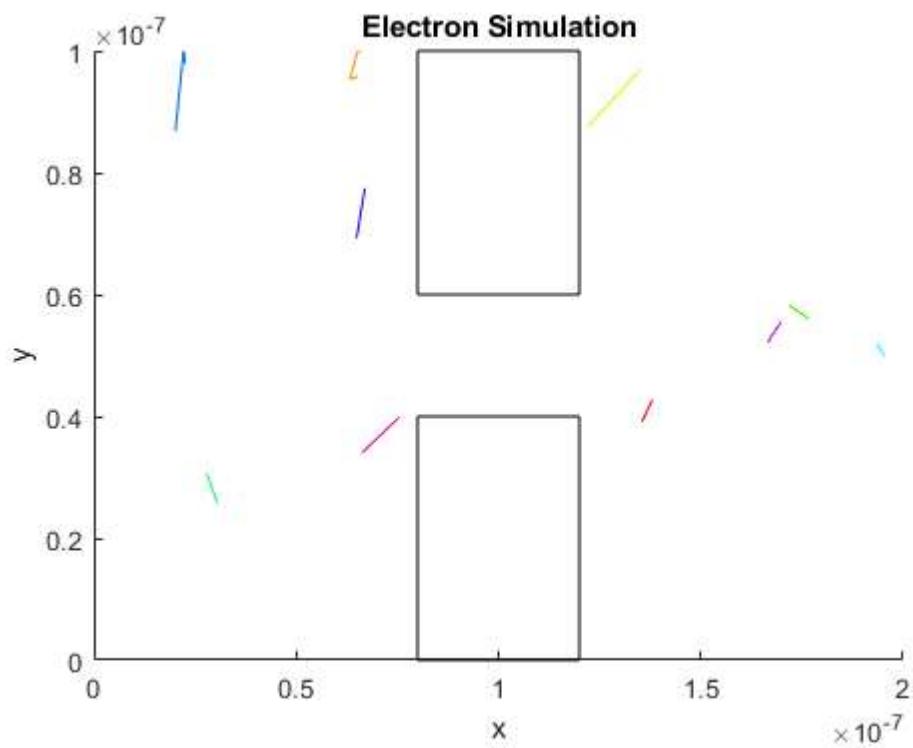
Electron Simulation

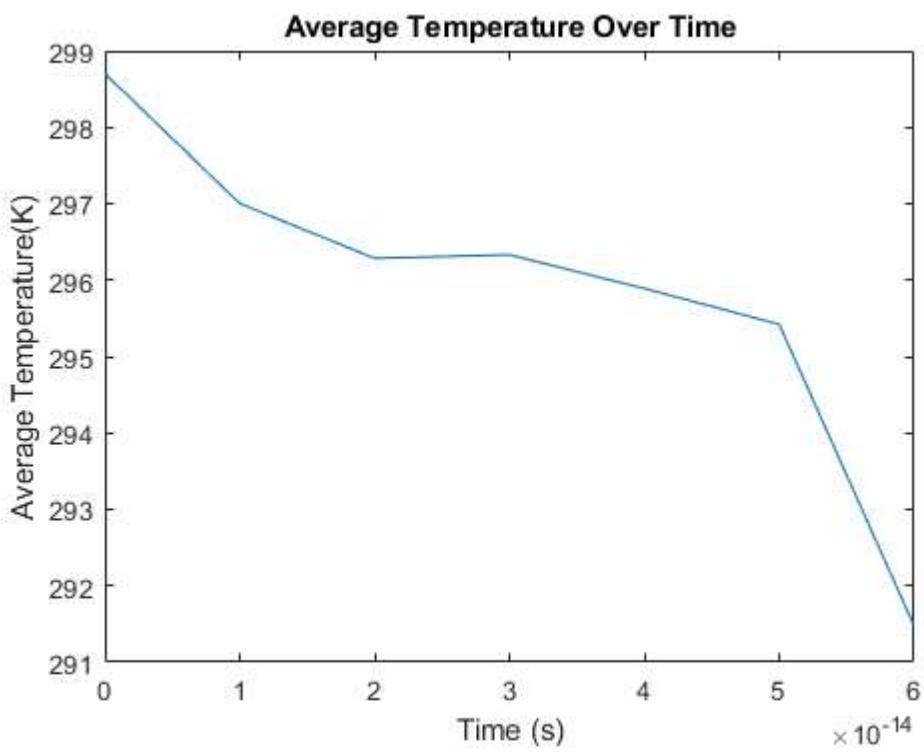
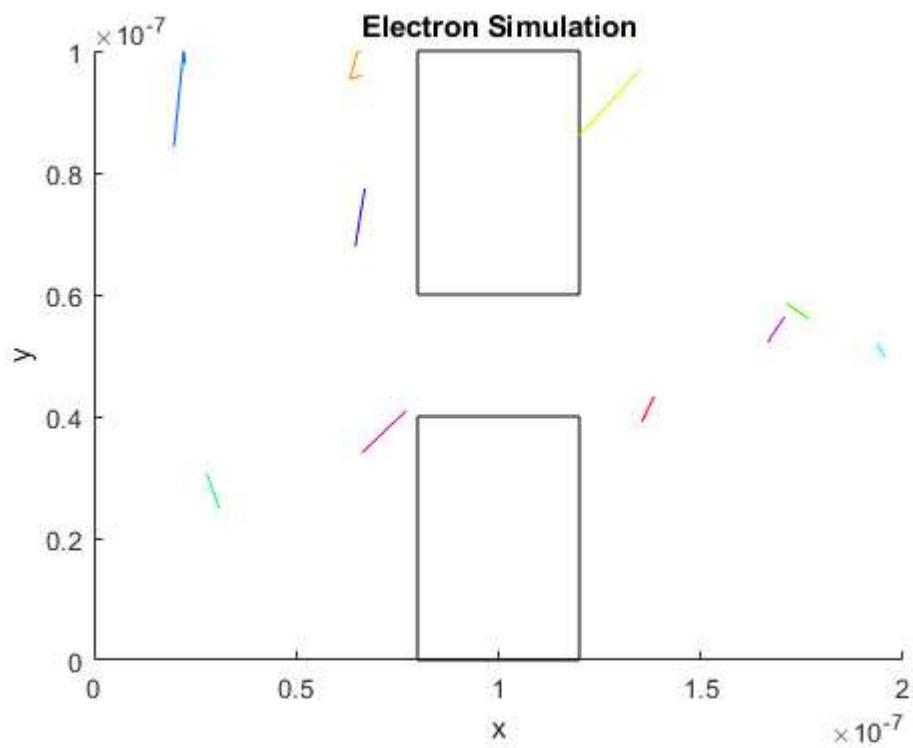


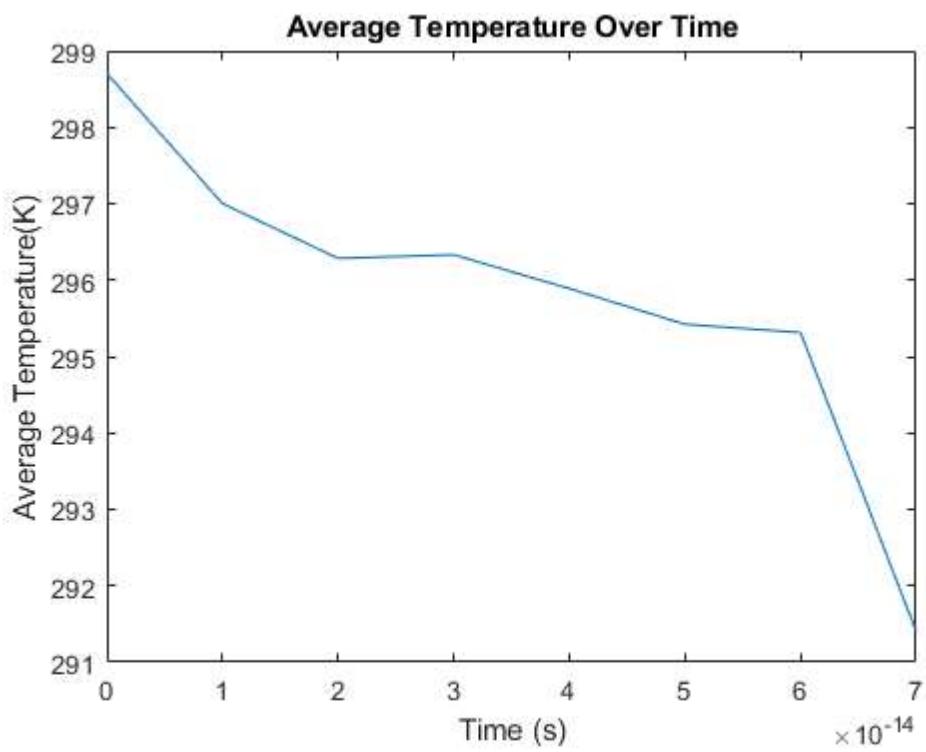
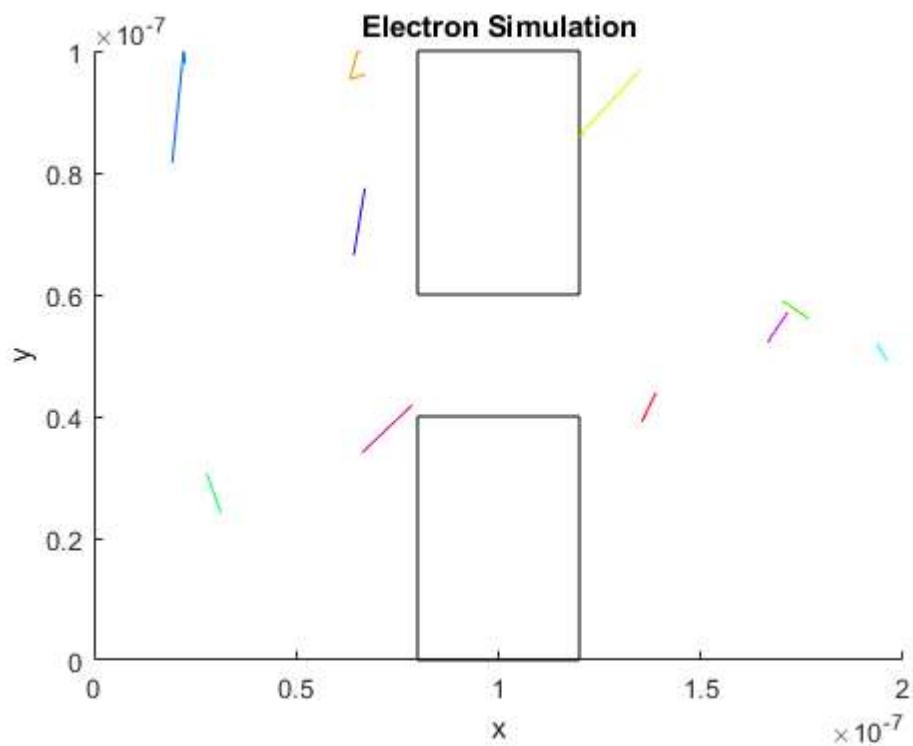


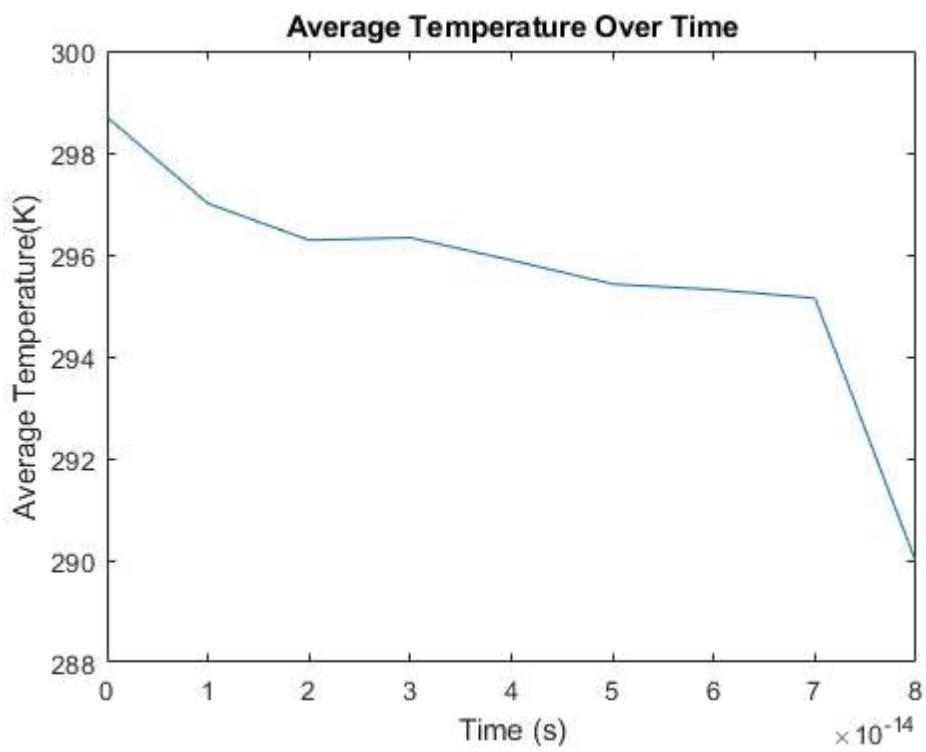
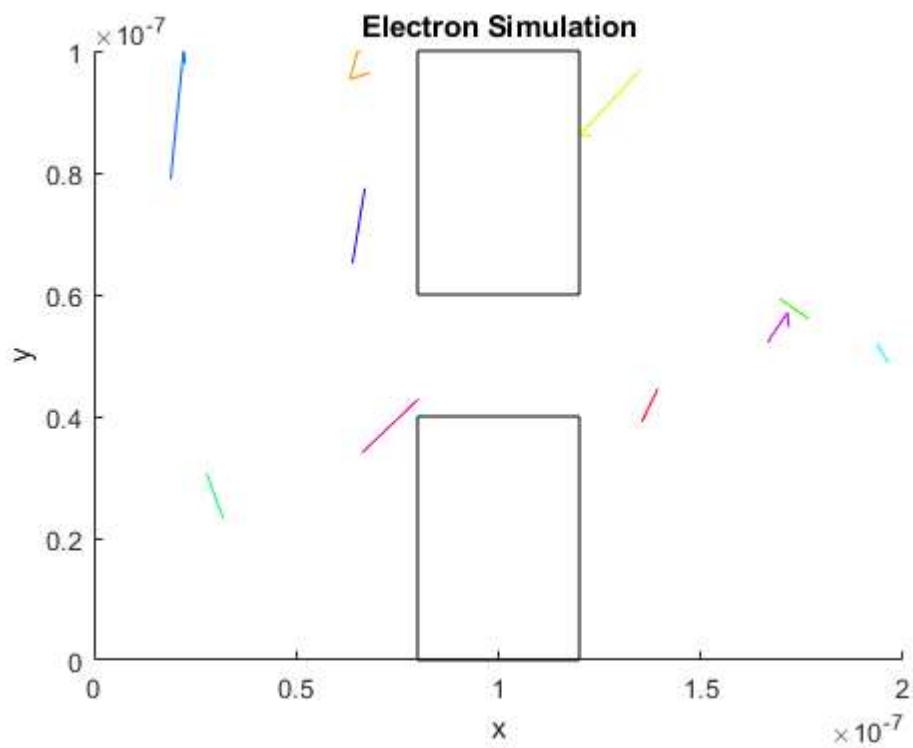


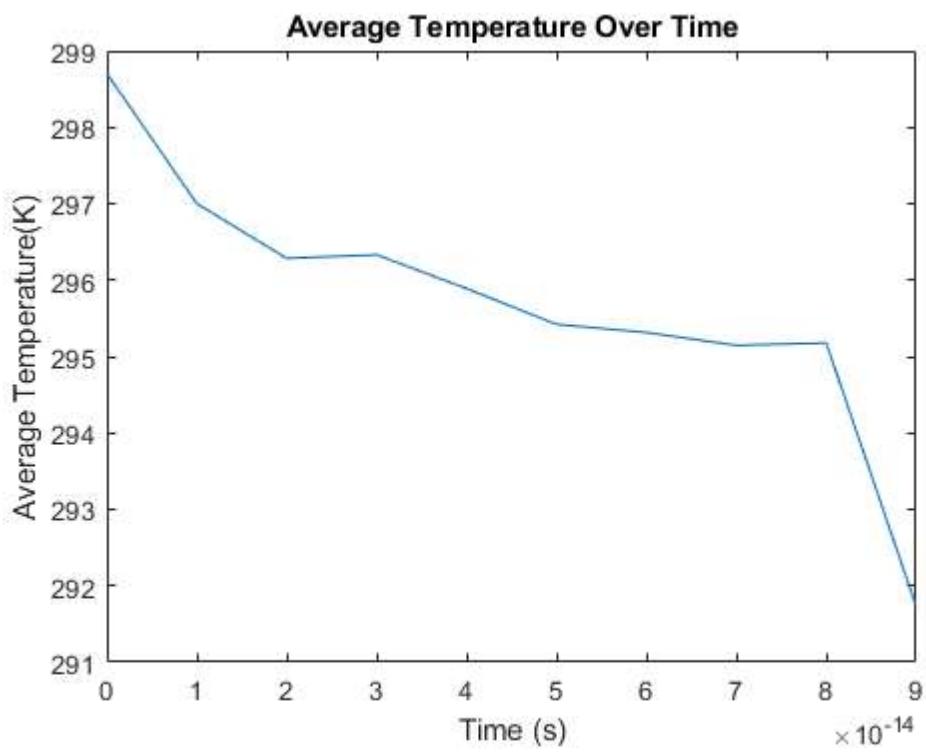
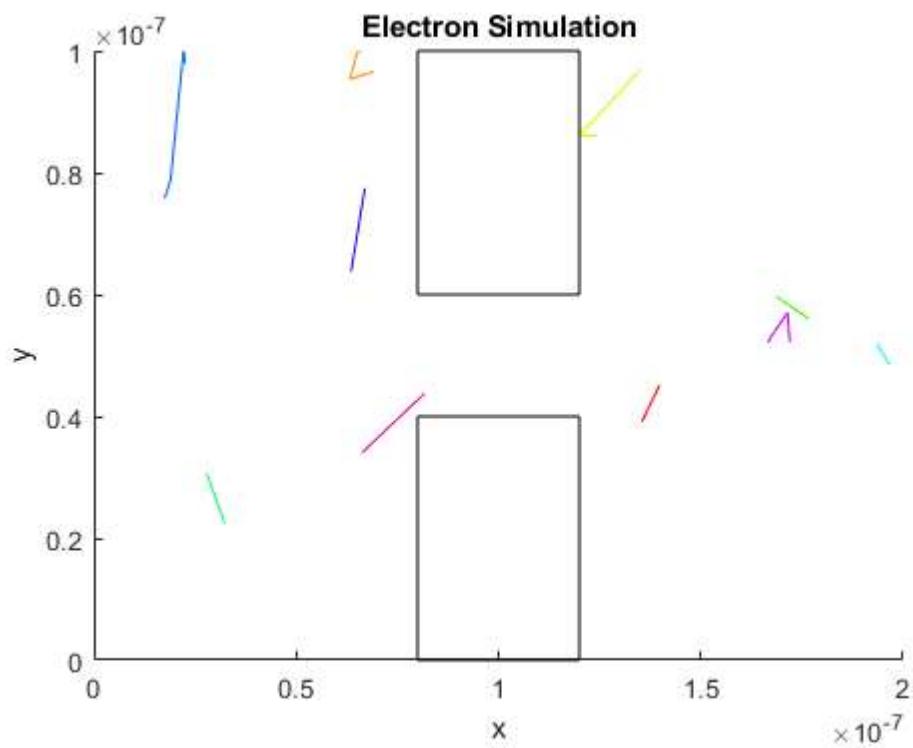


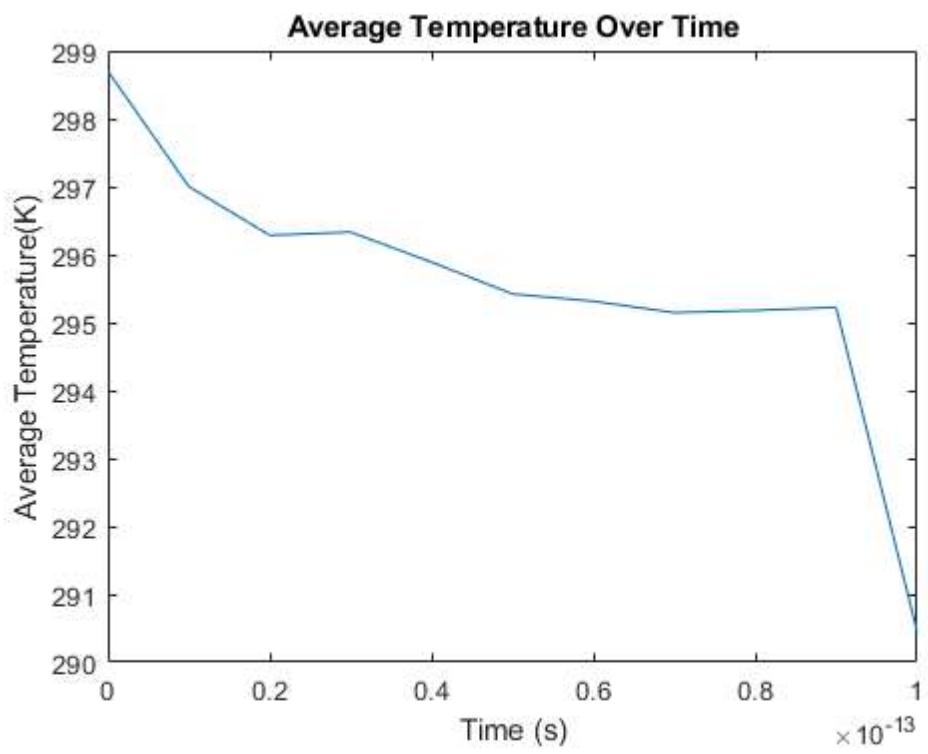
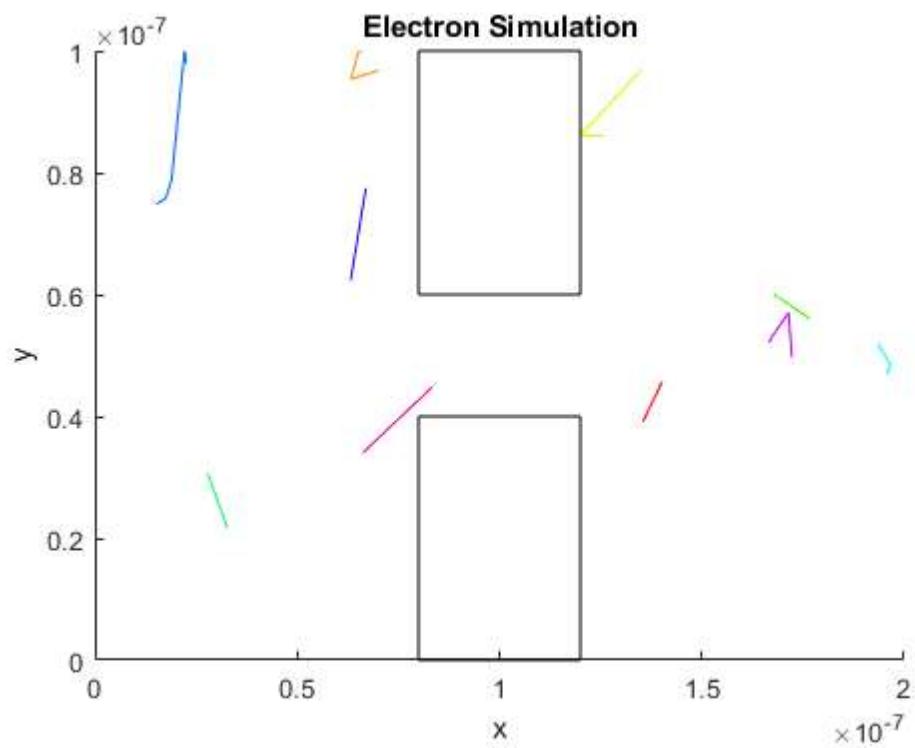


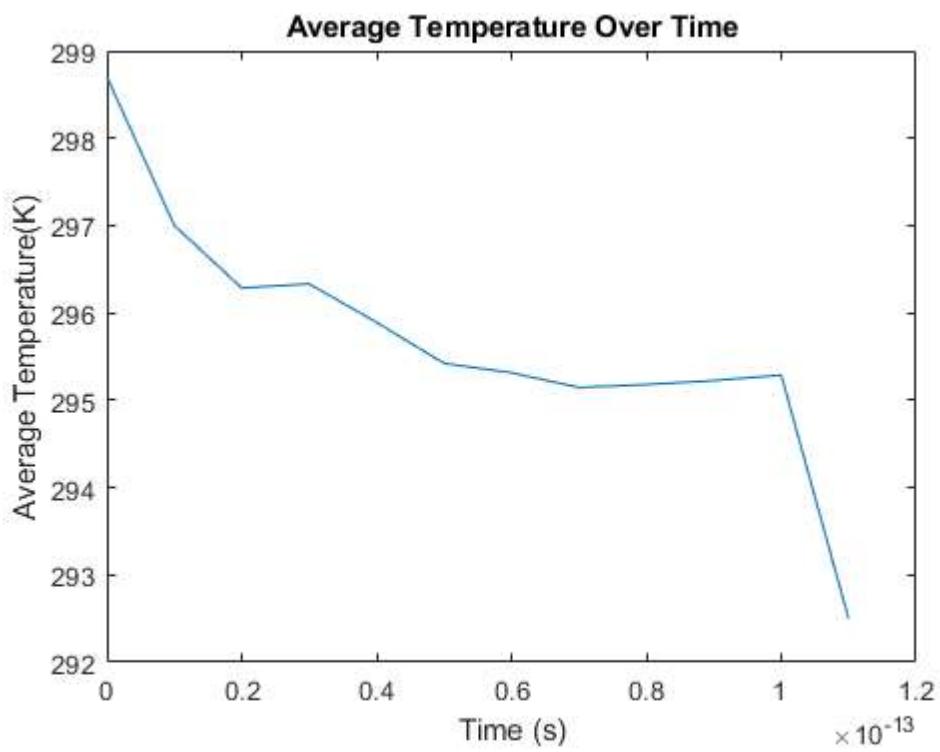
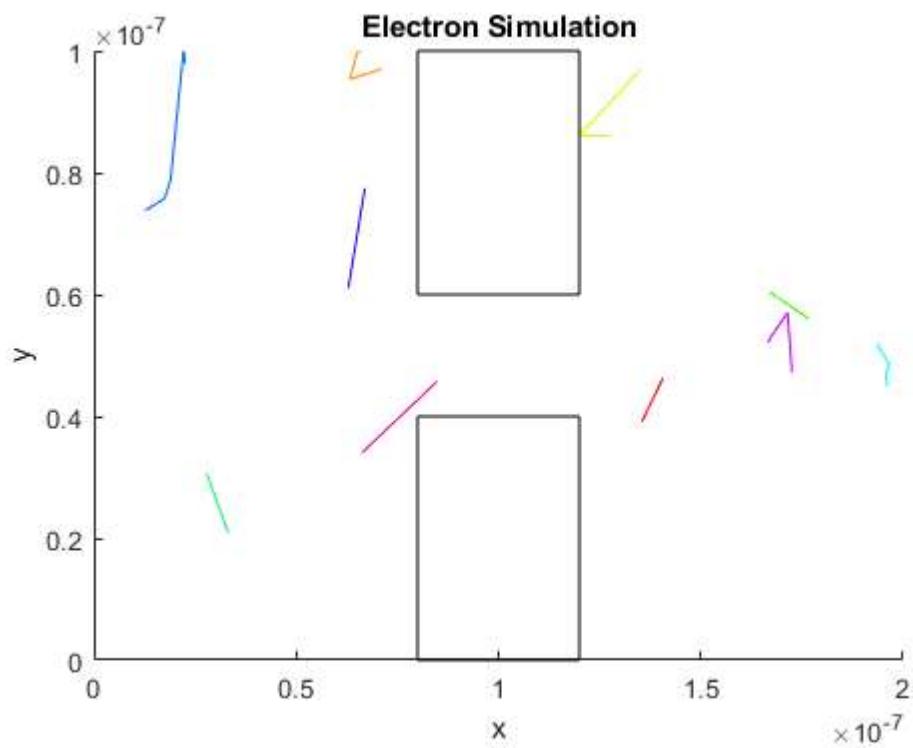


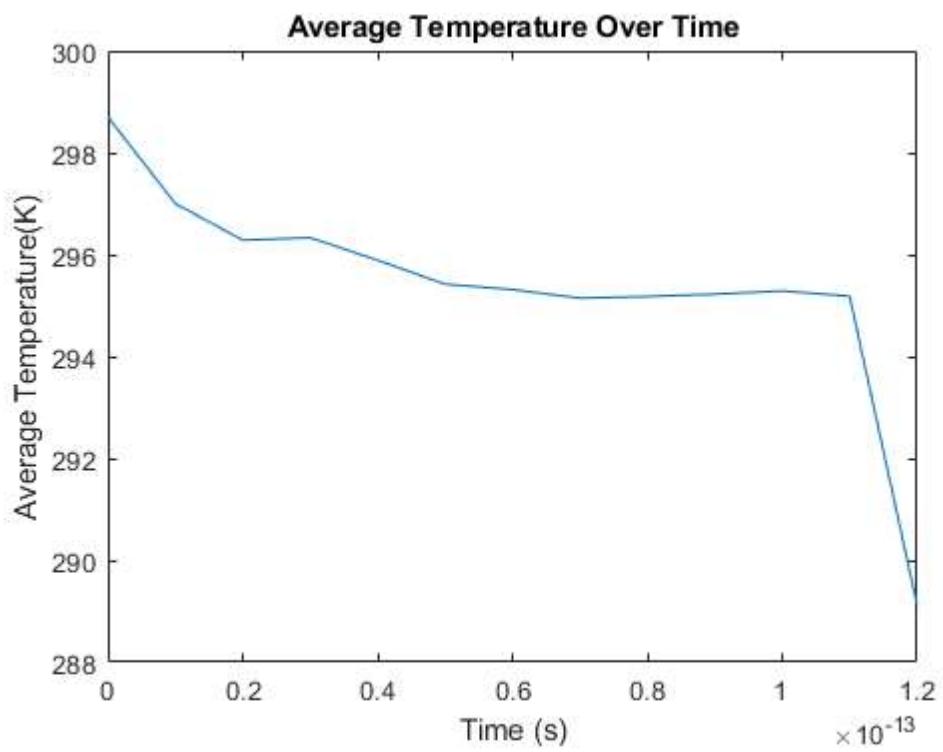
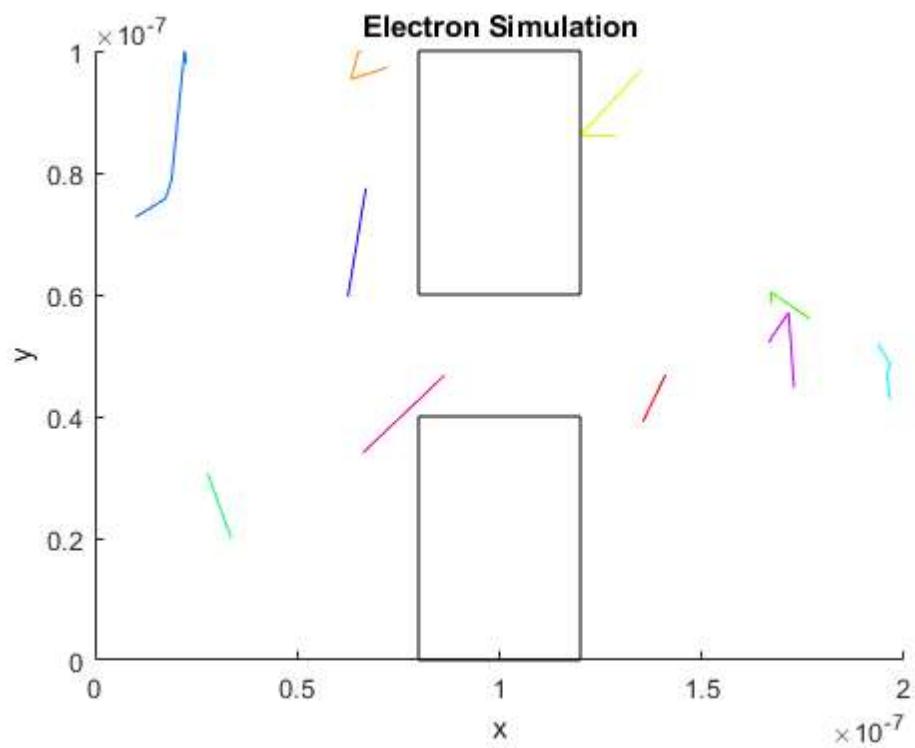


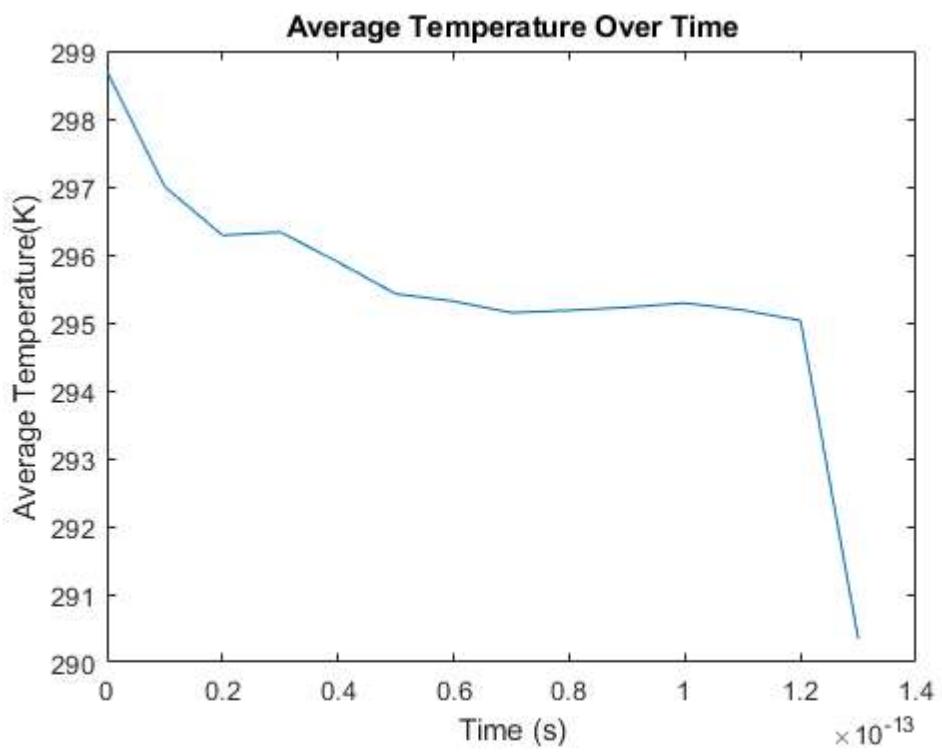
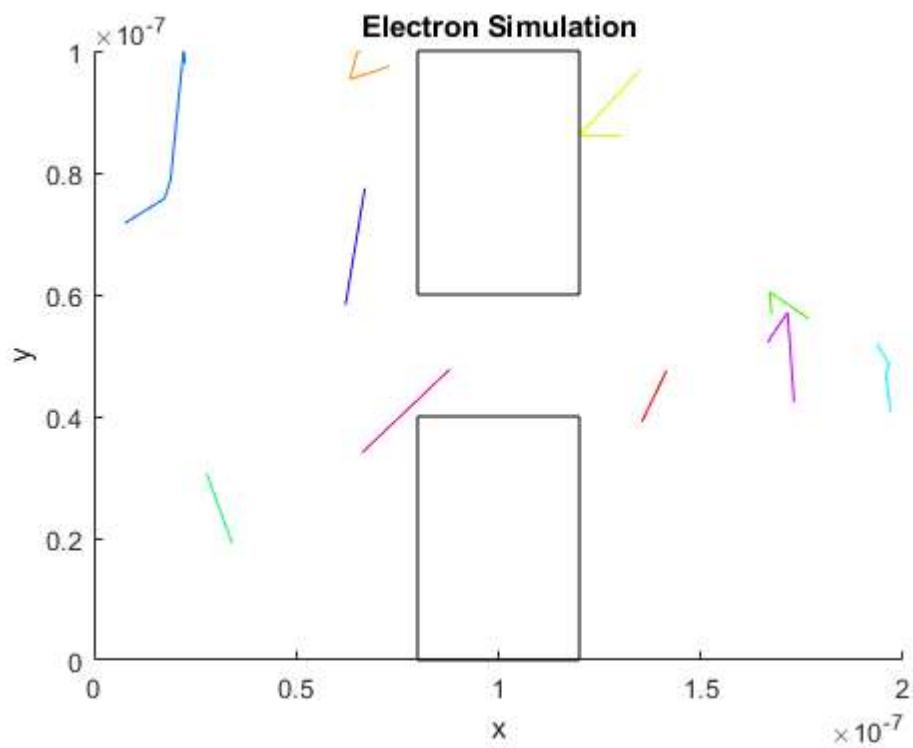


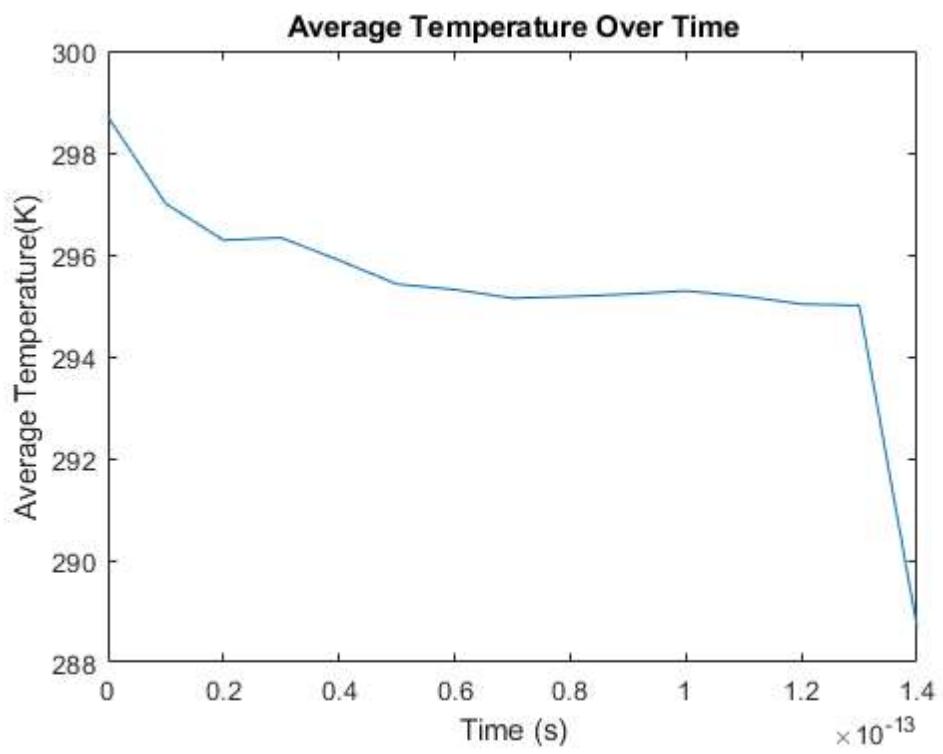
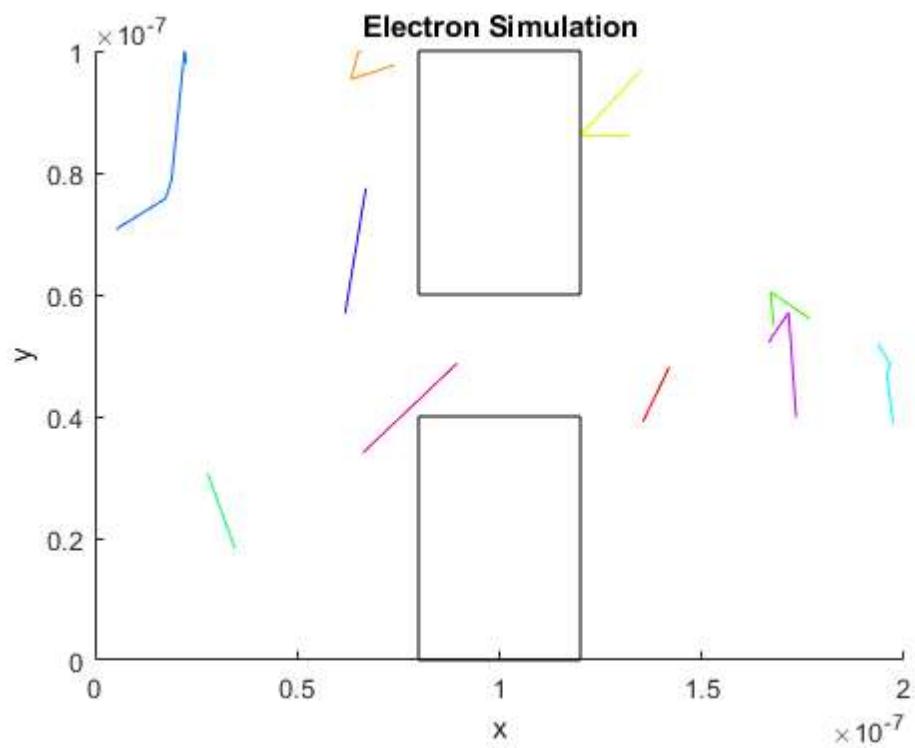


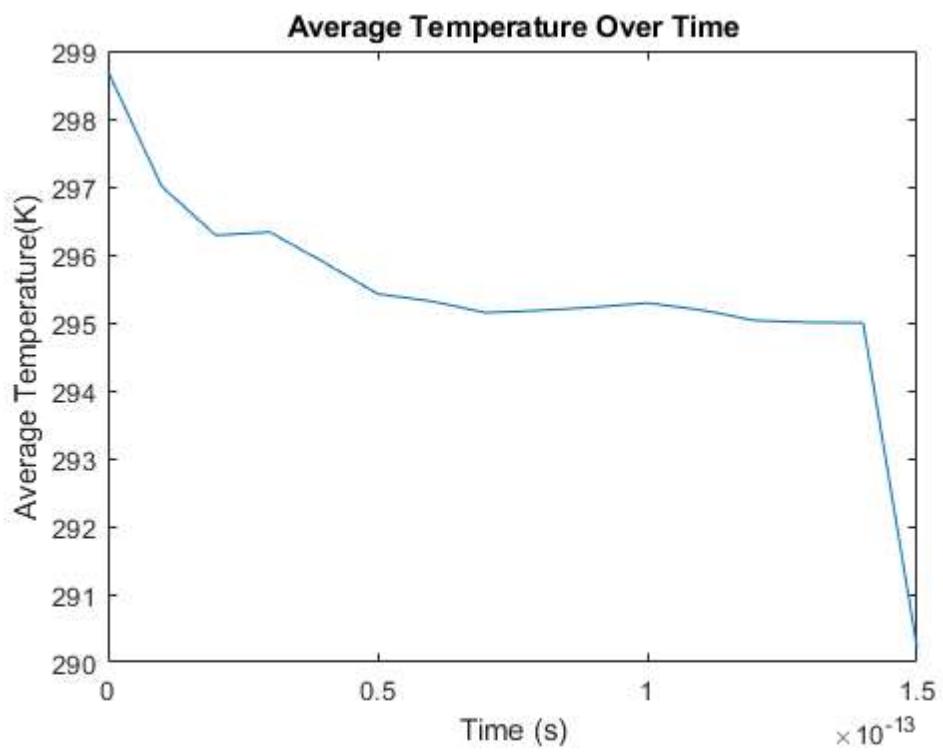
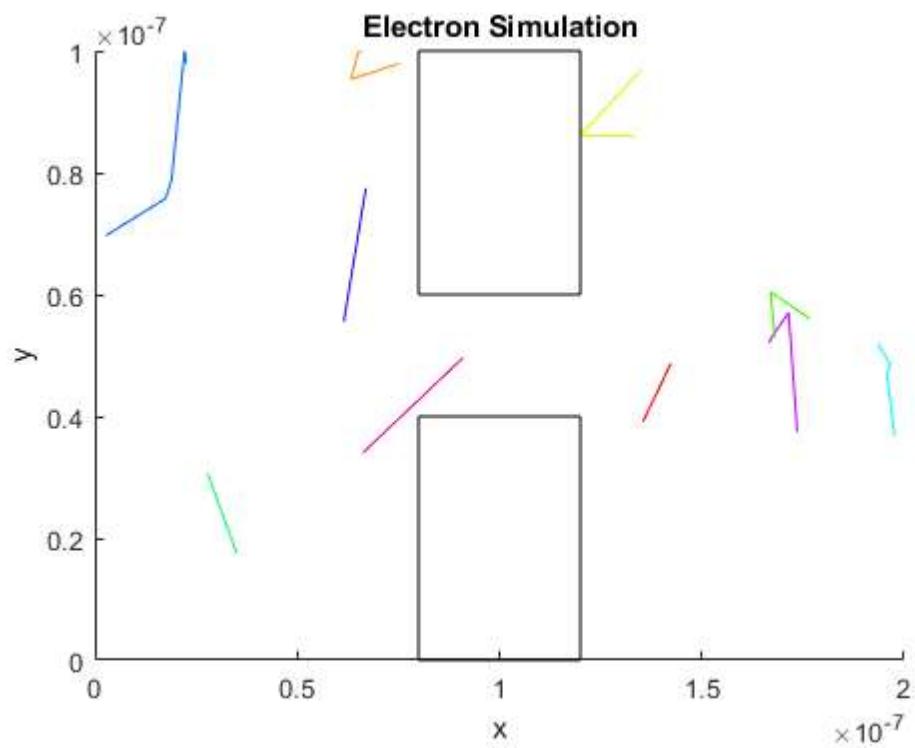


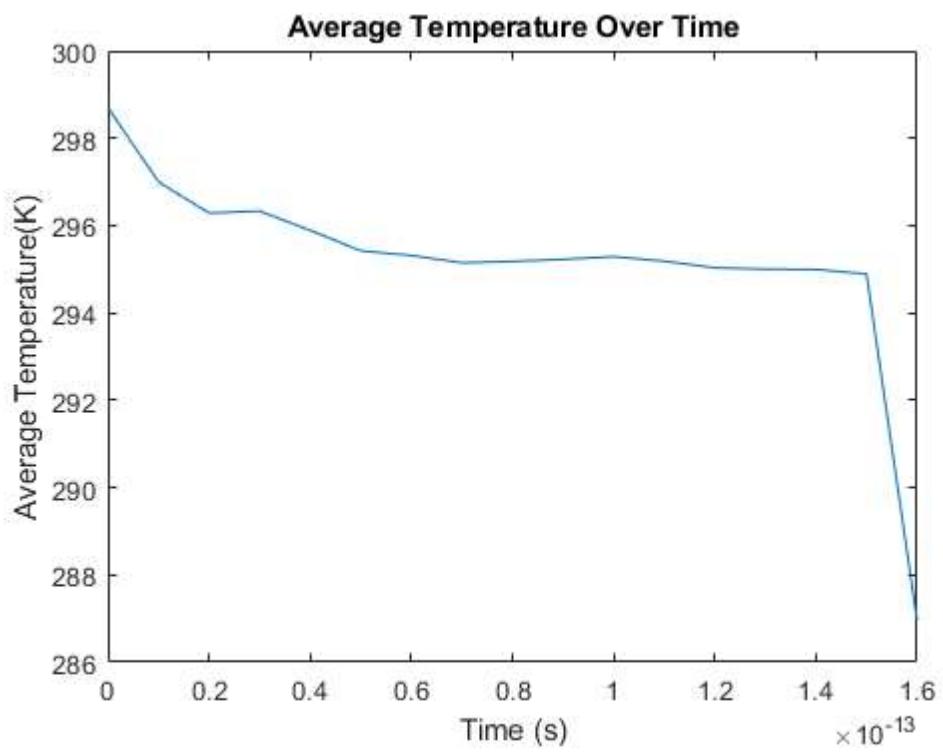
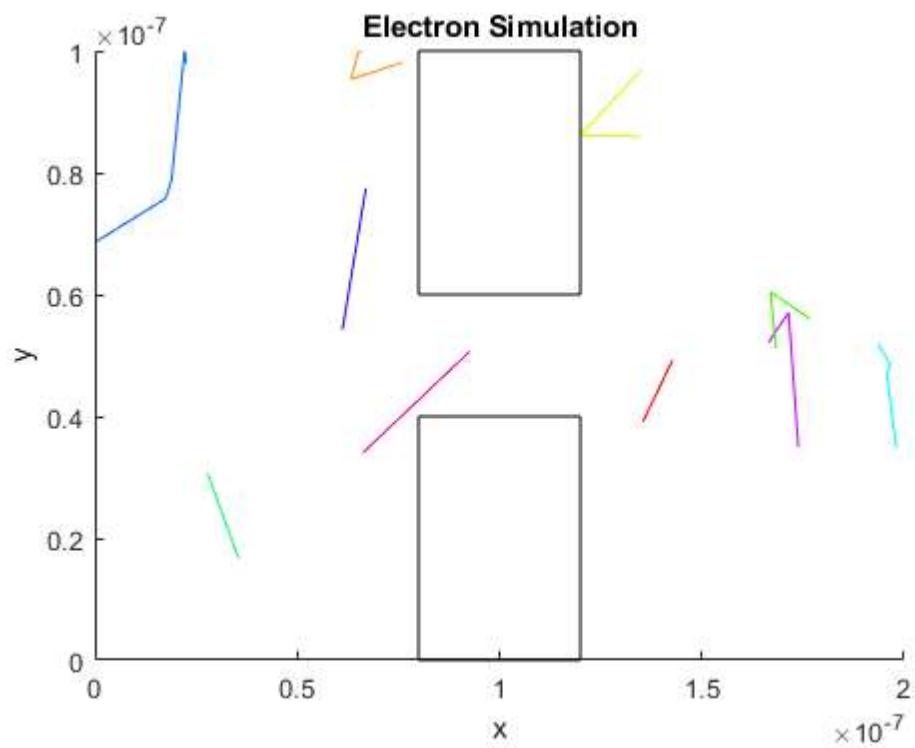


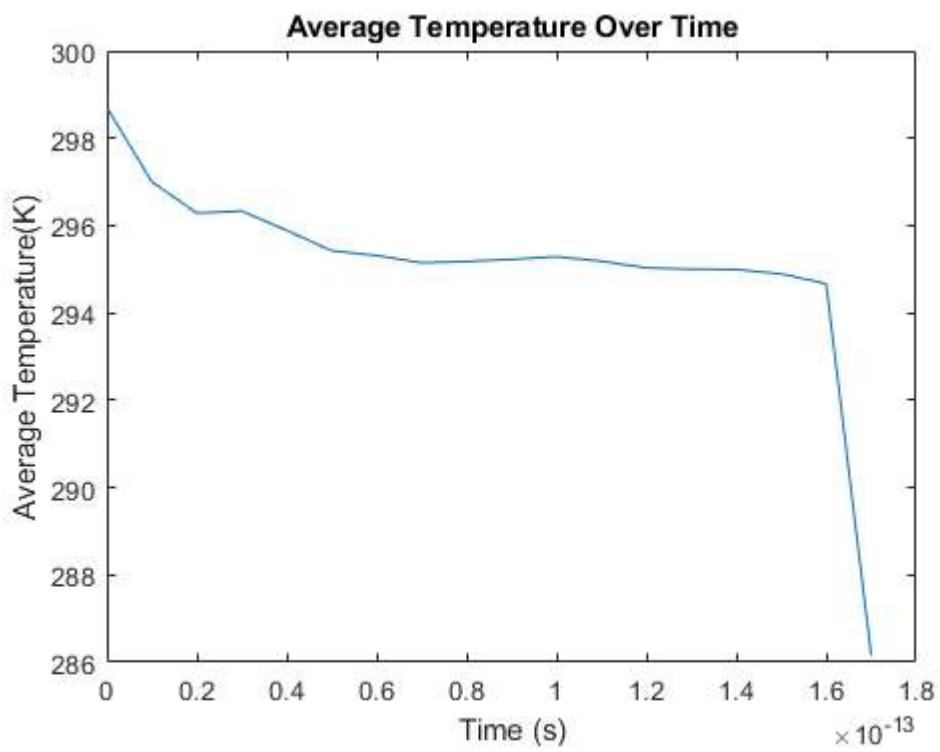
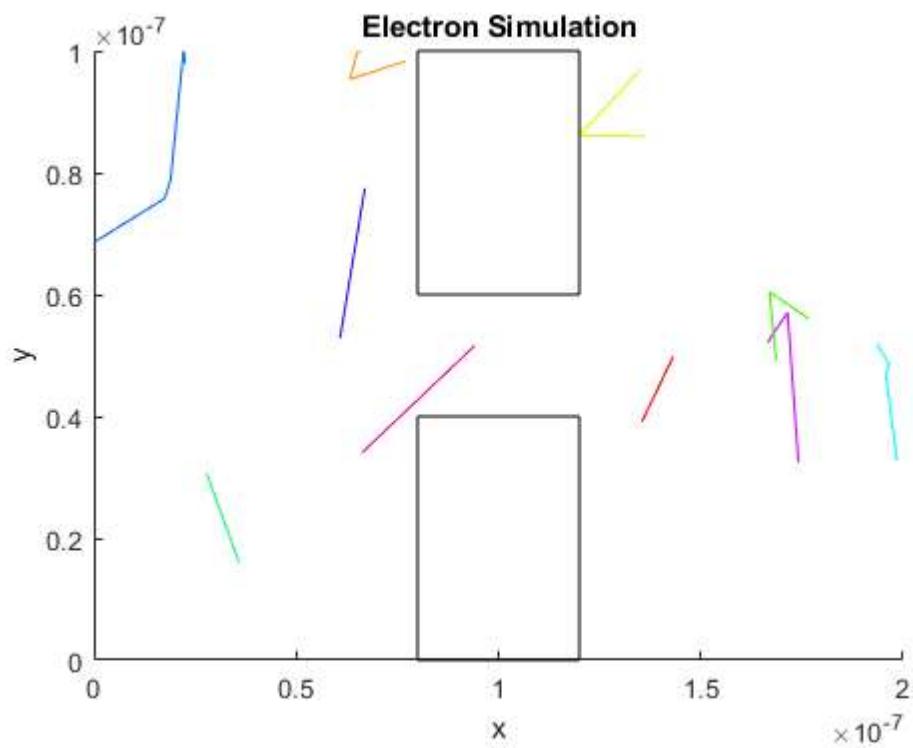


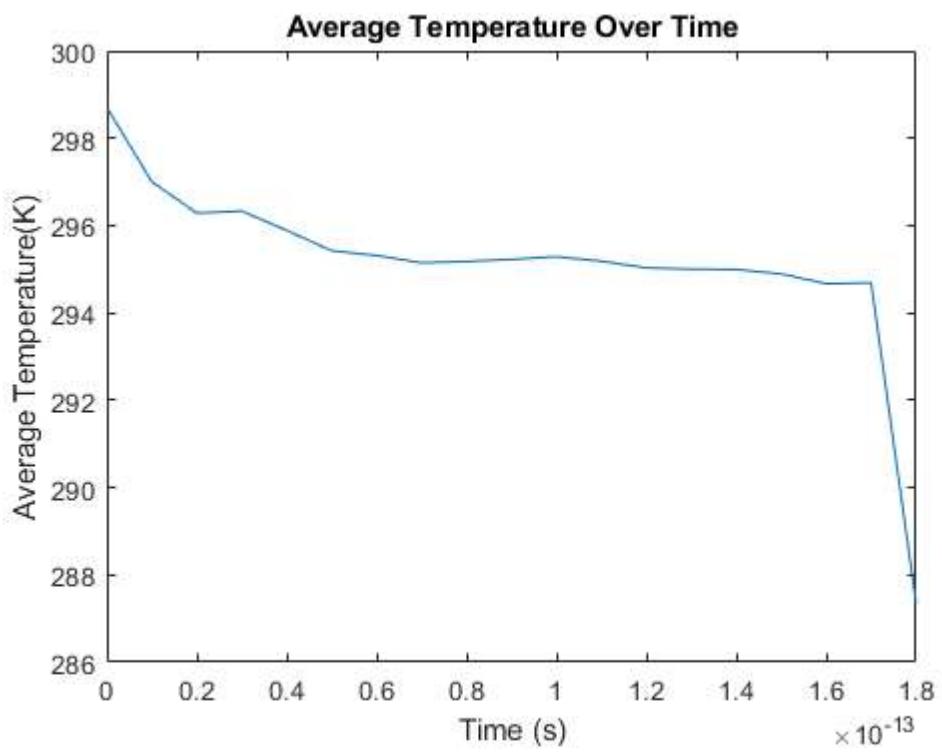
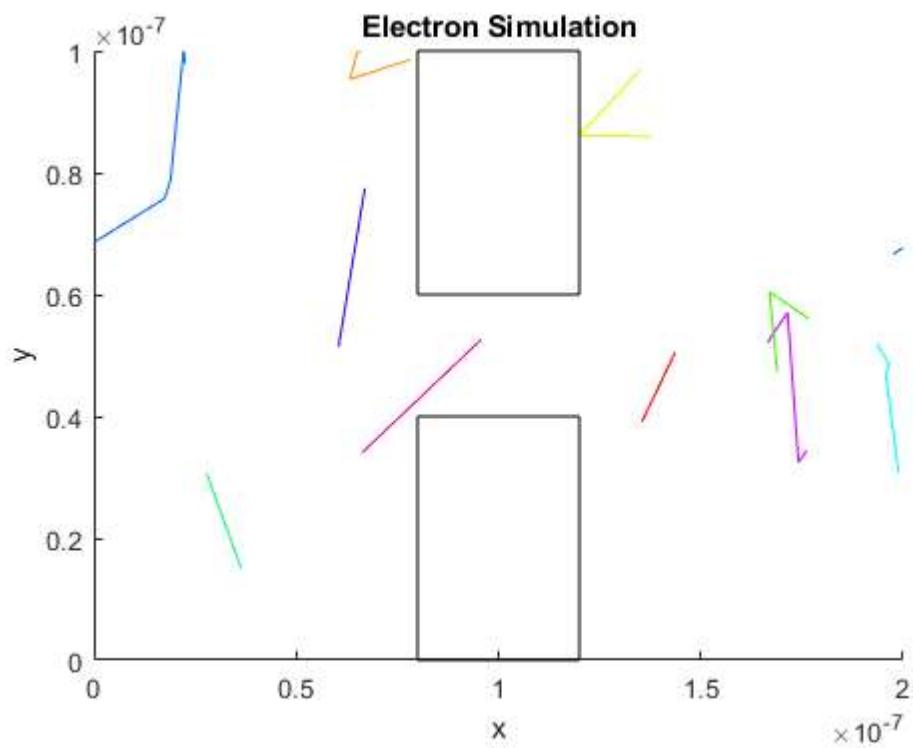


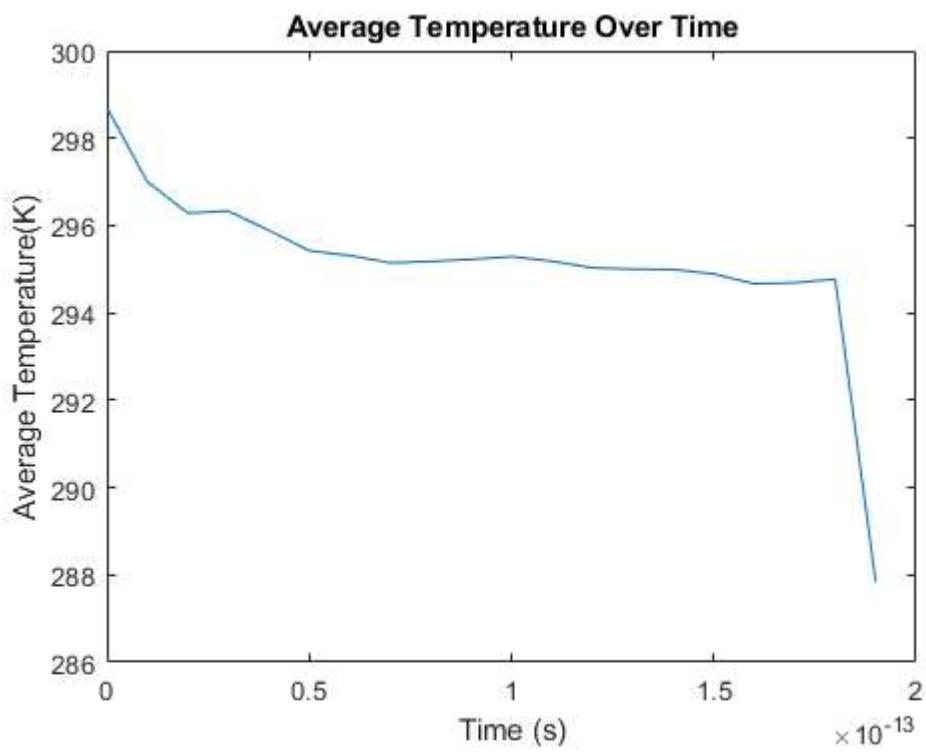
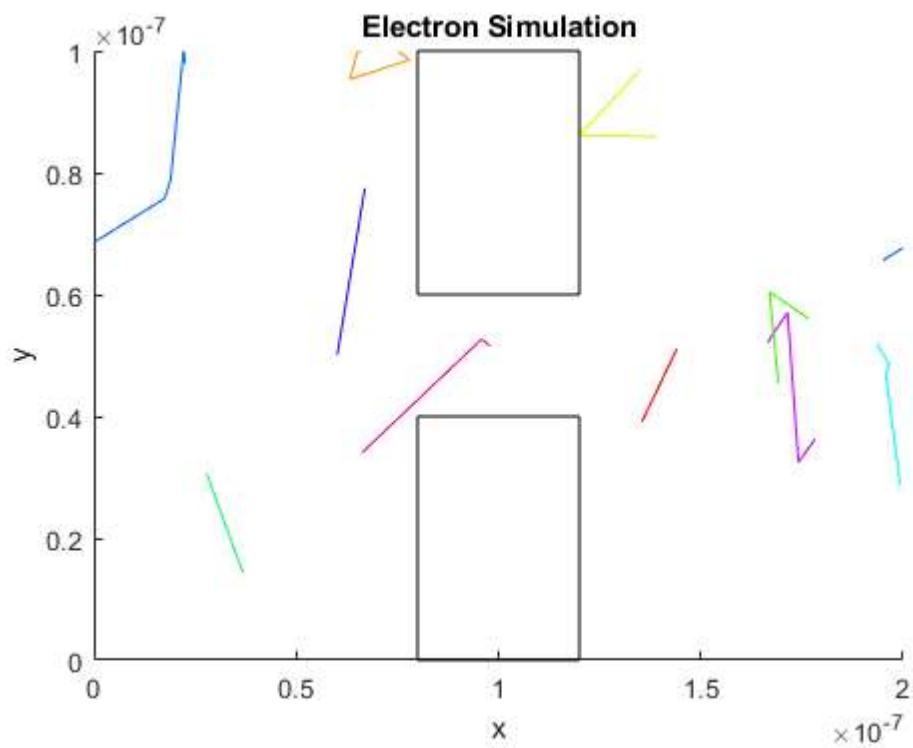


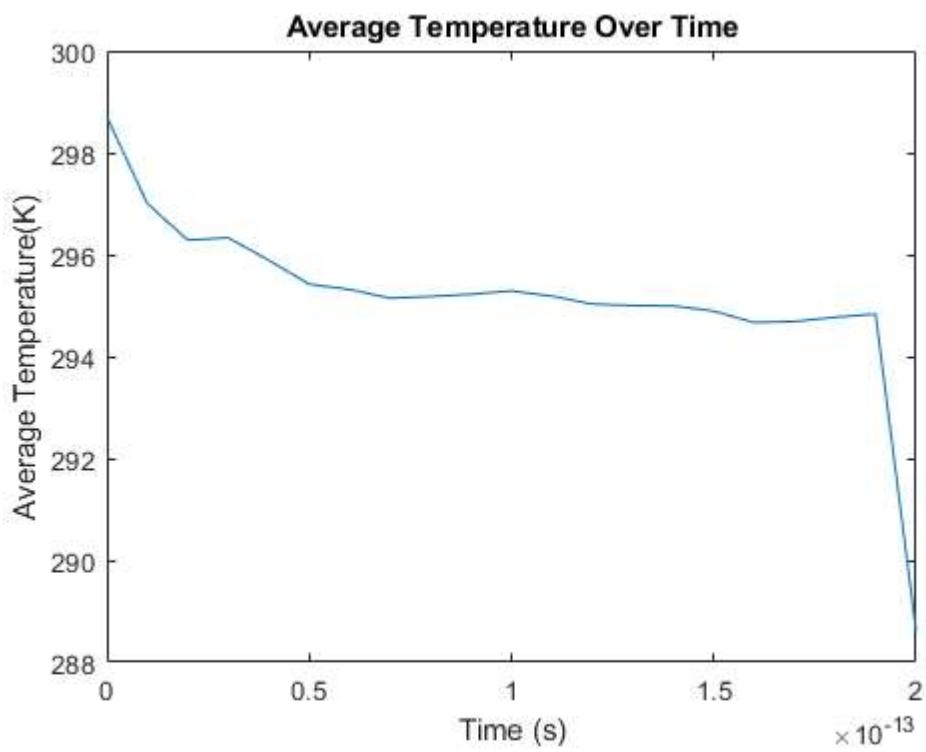
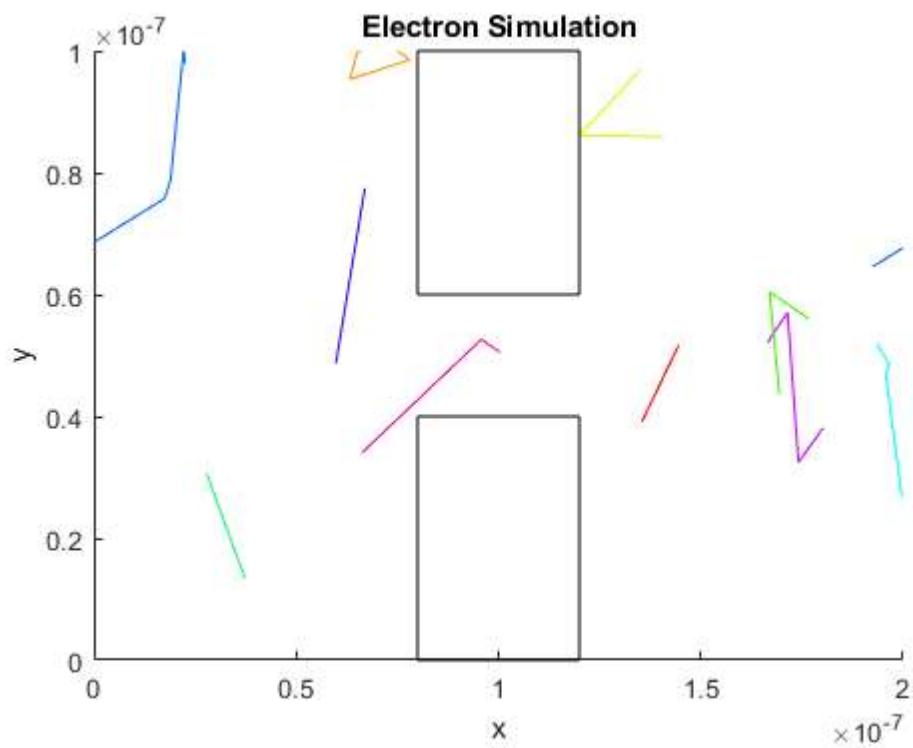


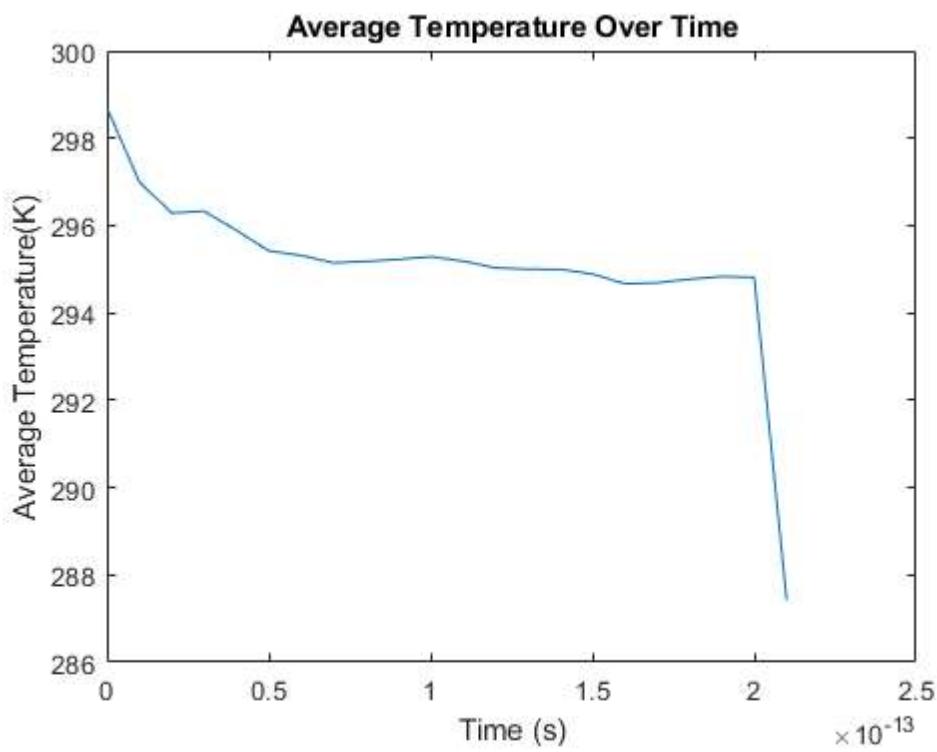
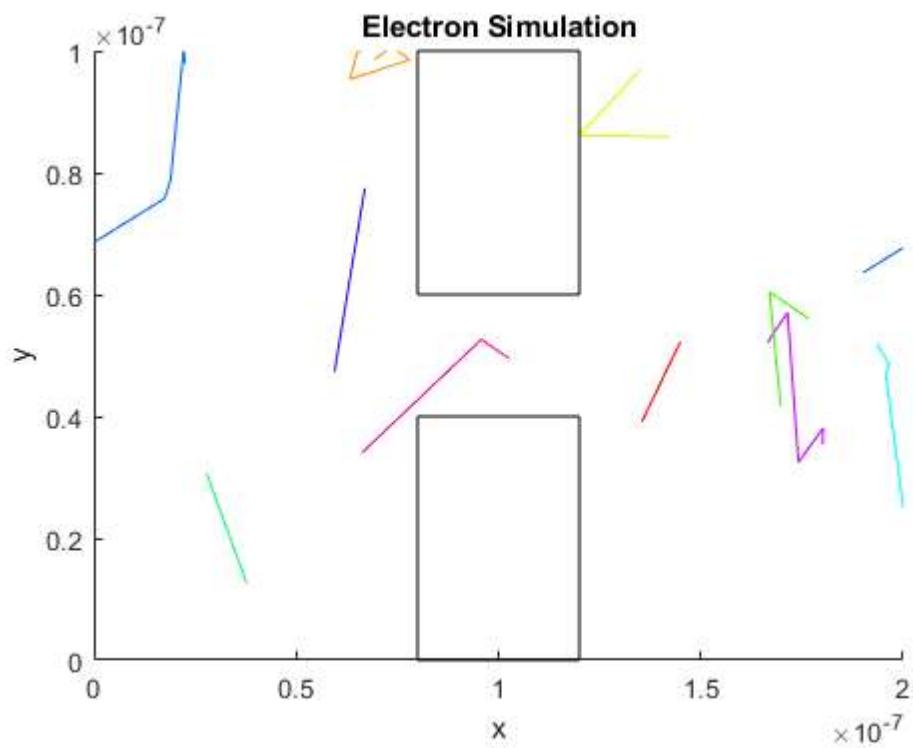


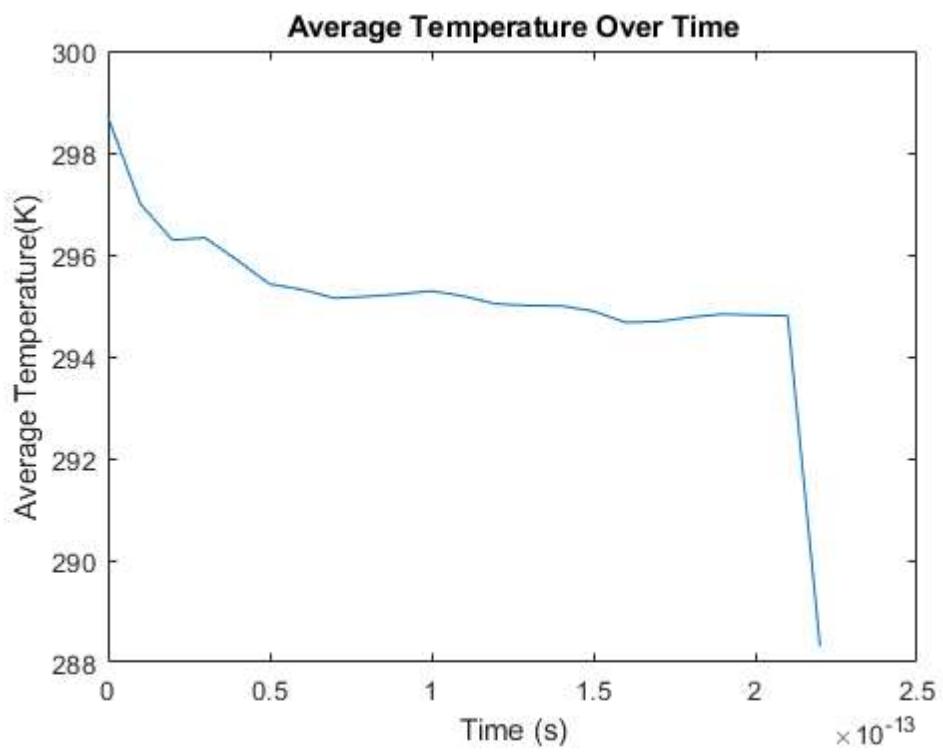
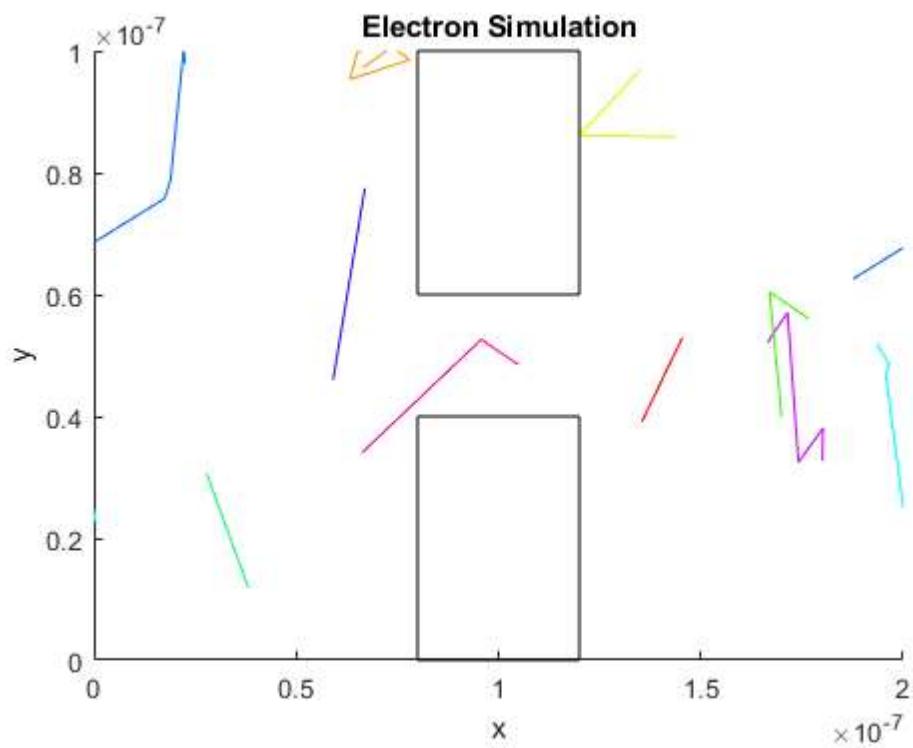


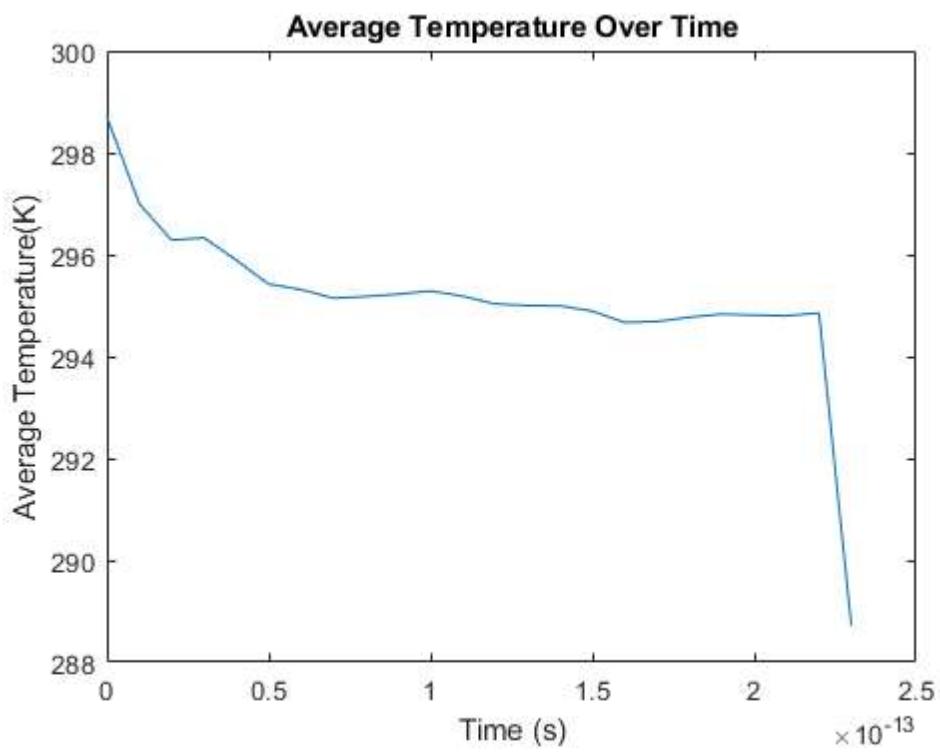
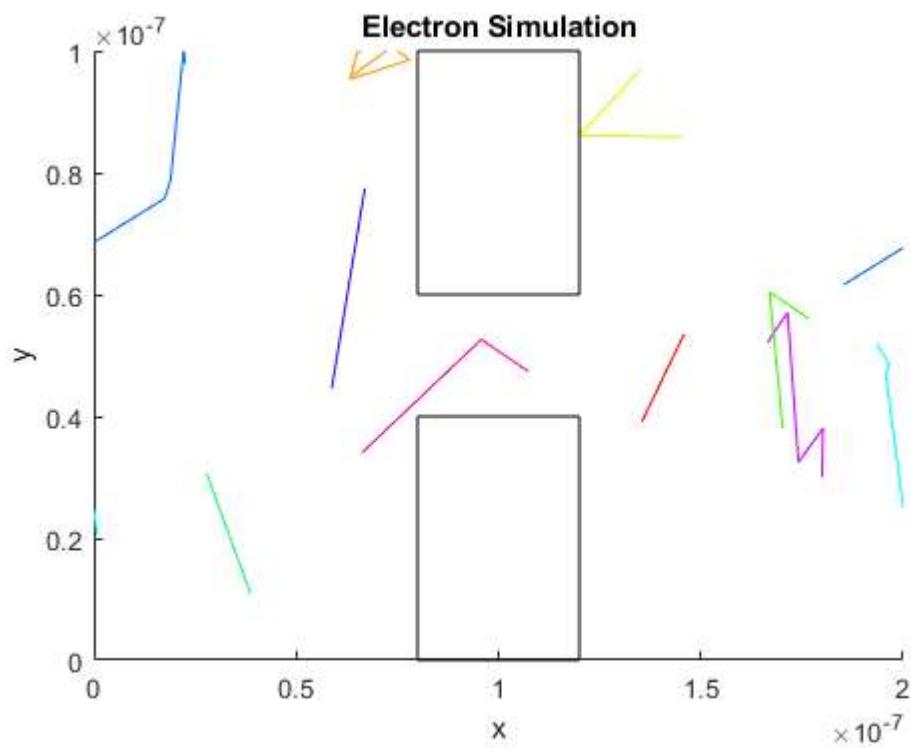


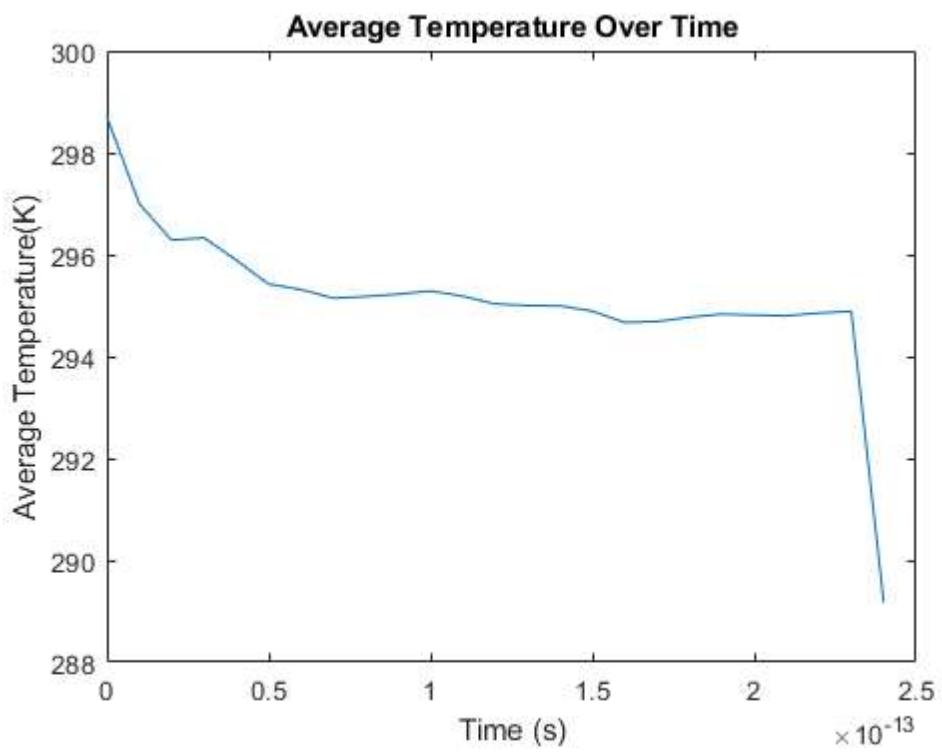
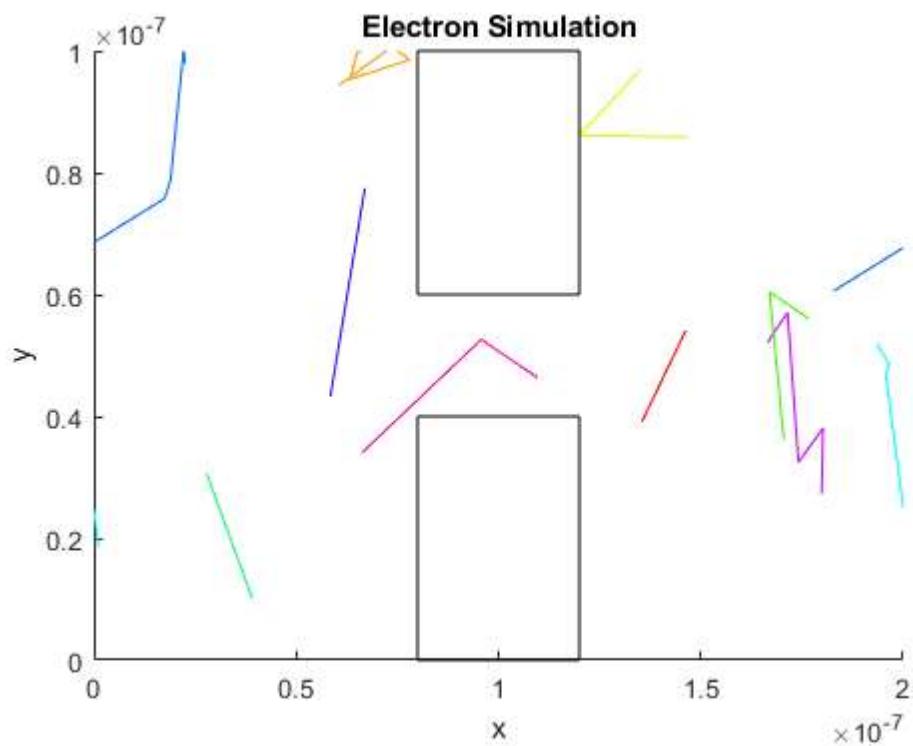


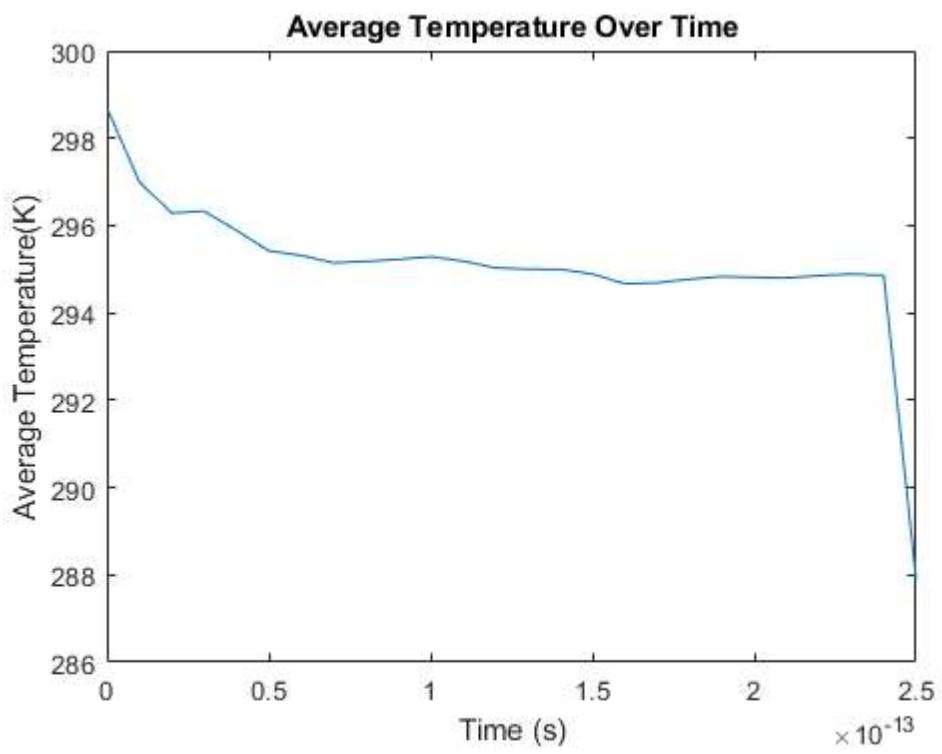
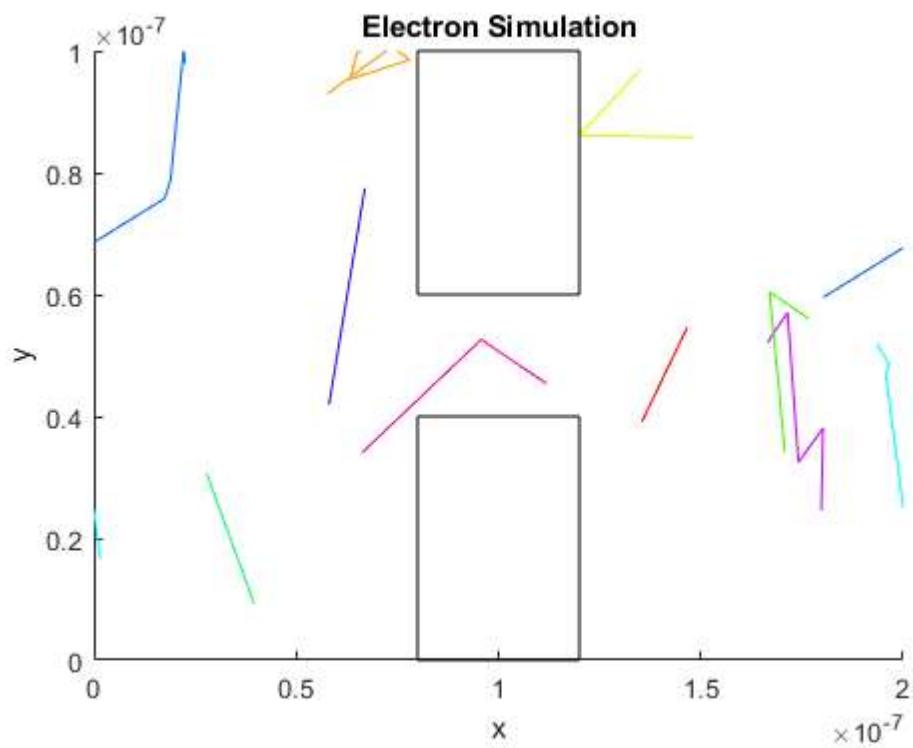


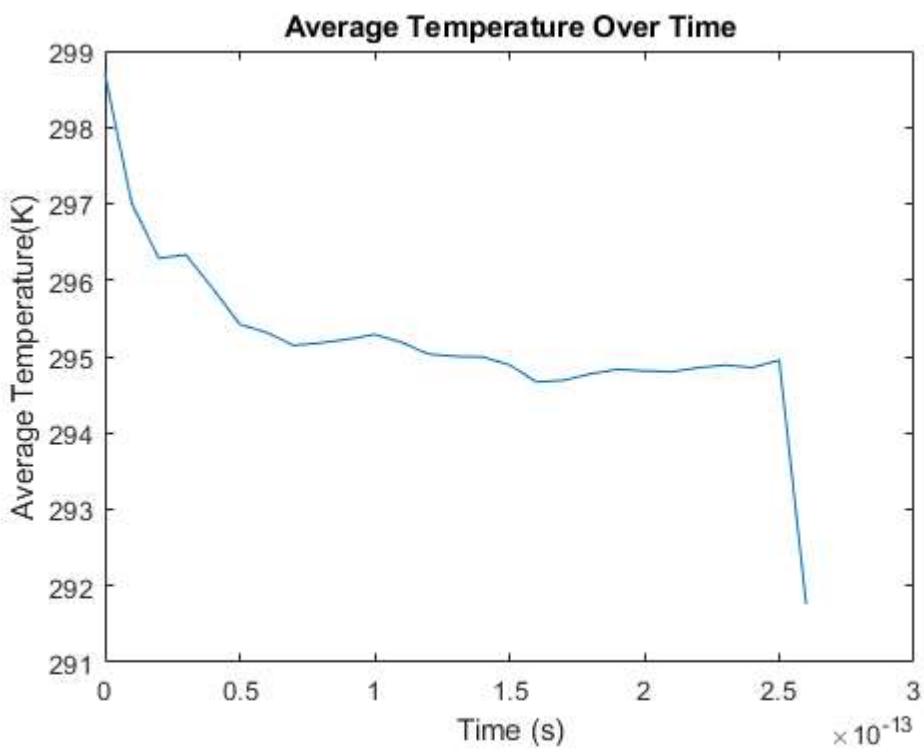
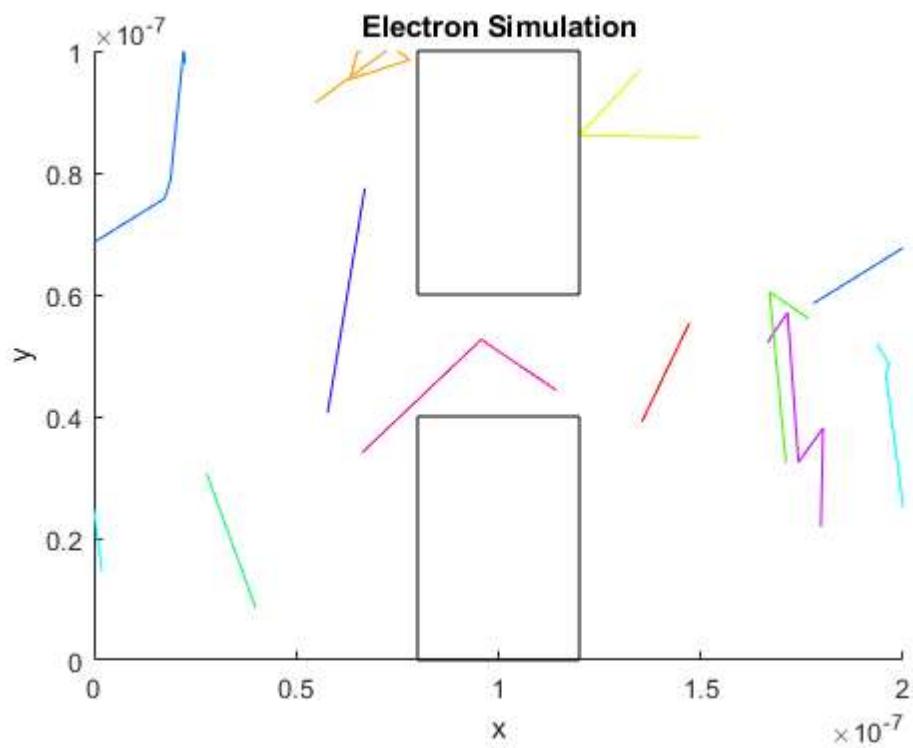


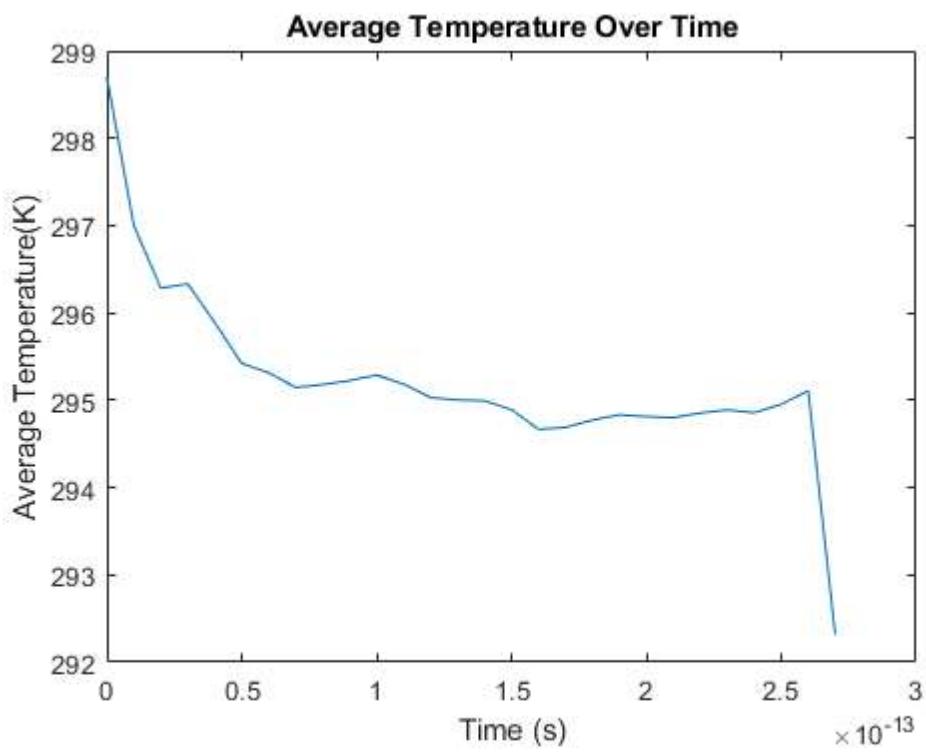
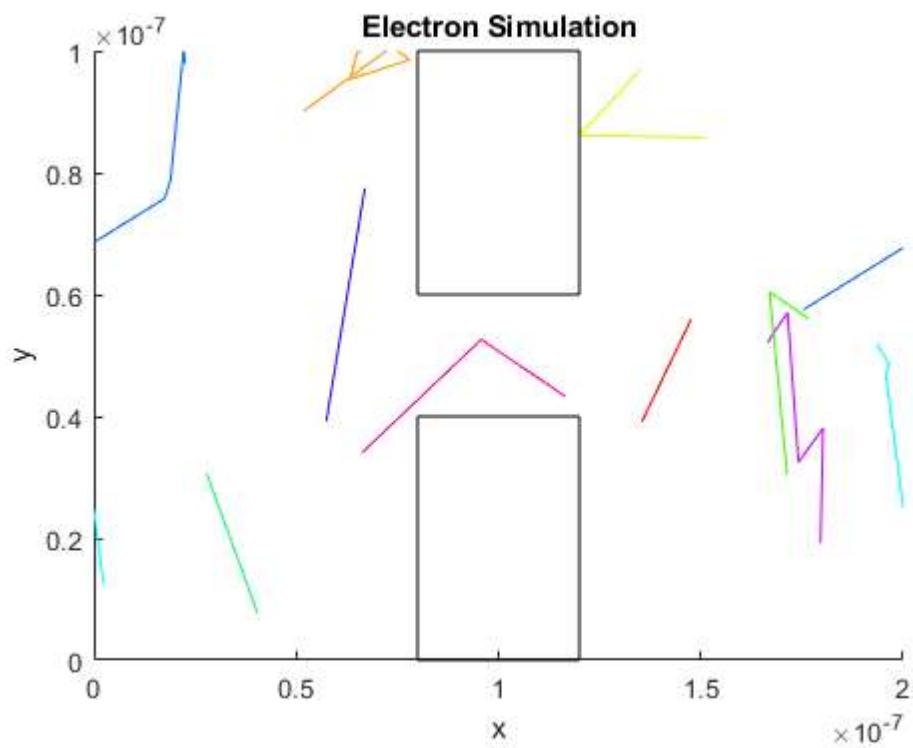


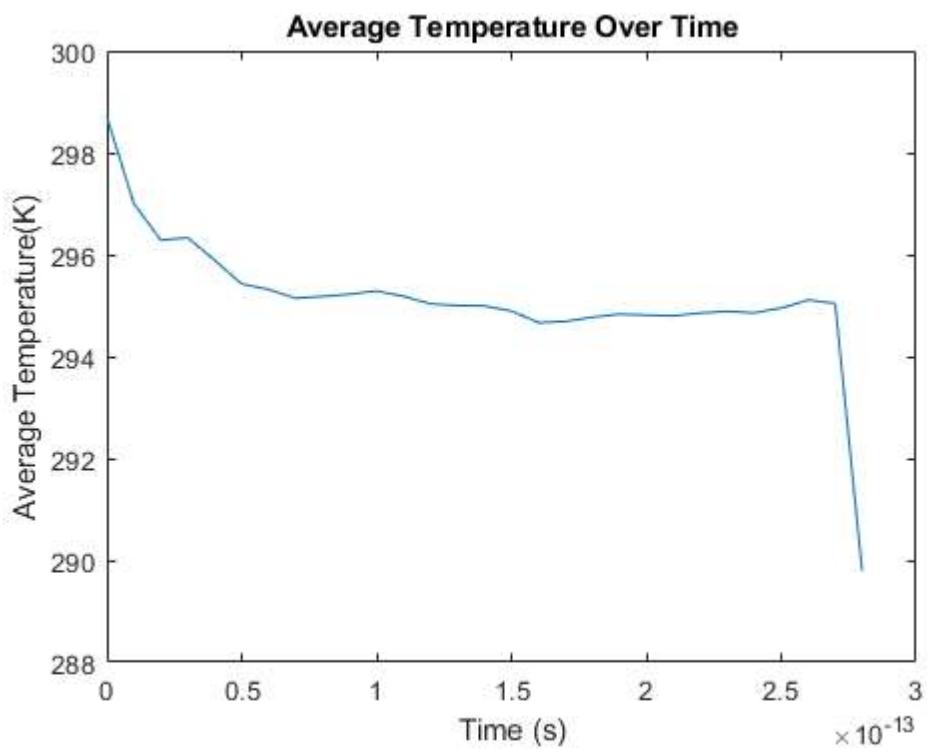
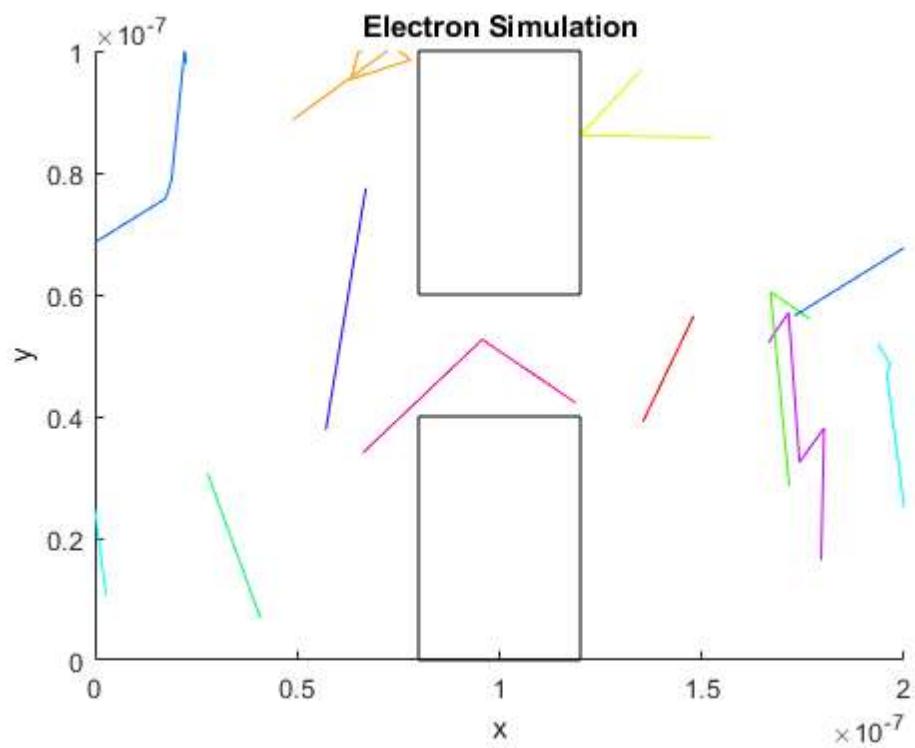


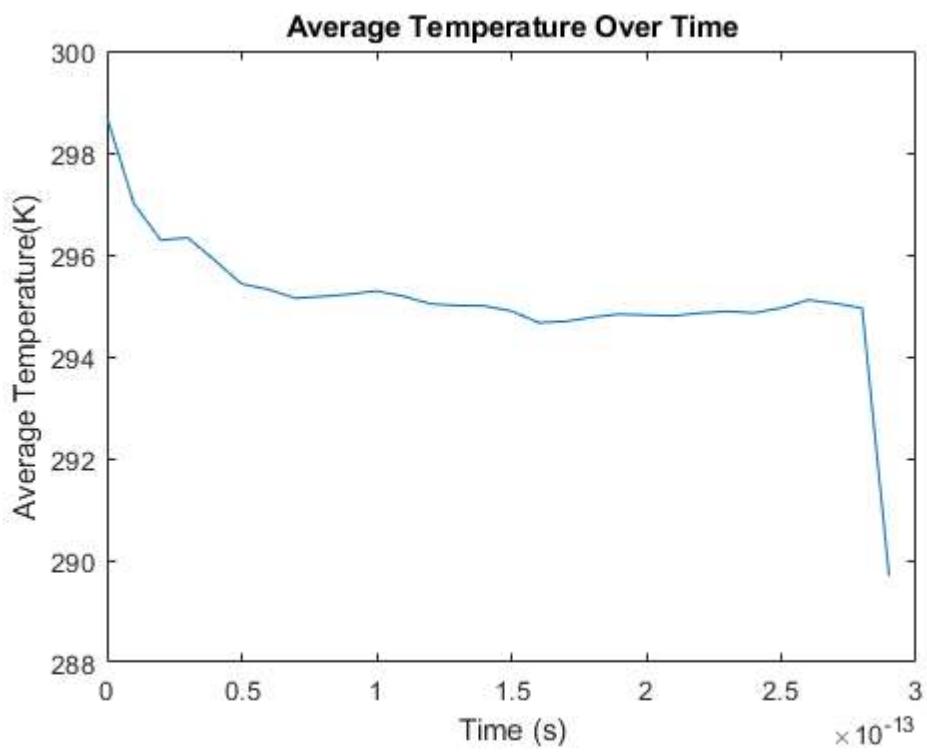
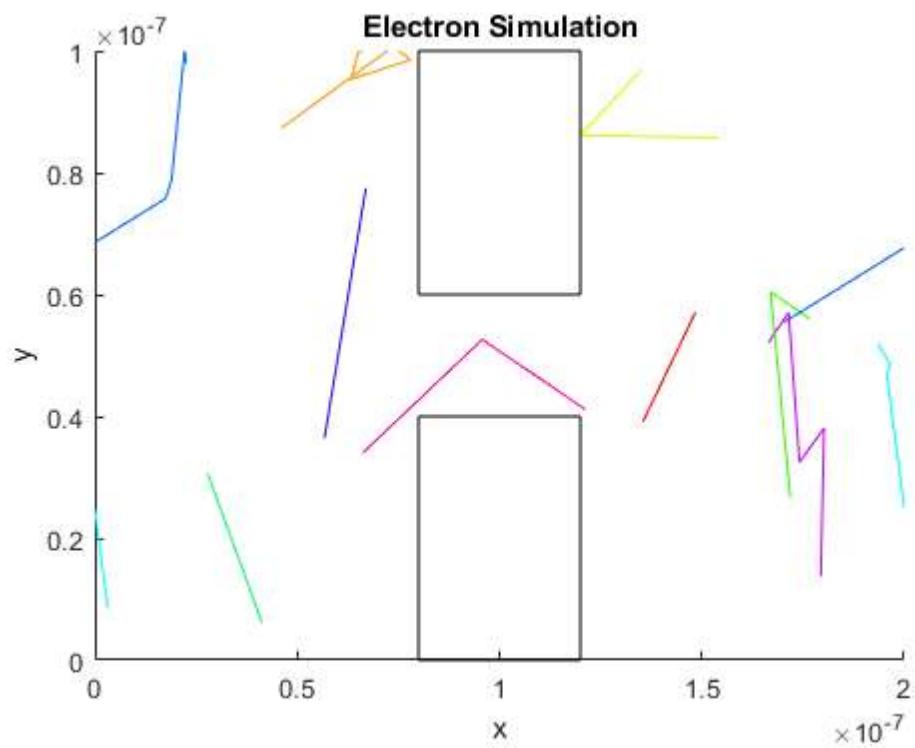


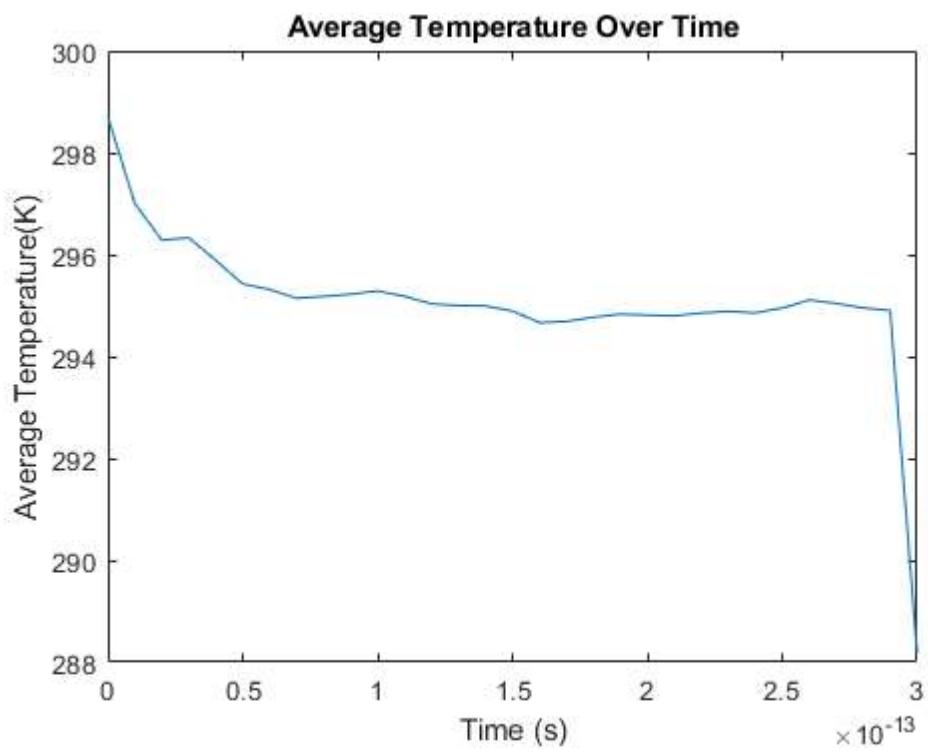
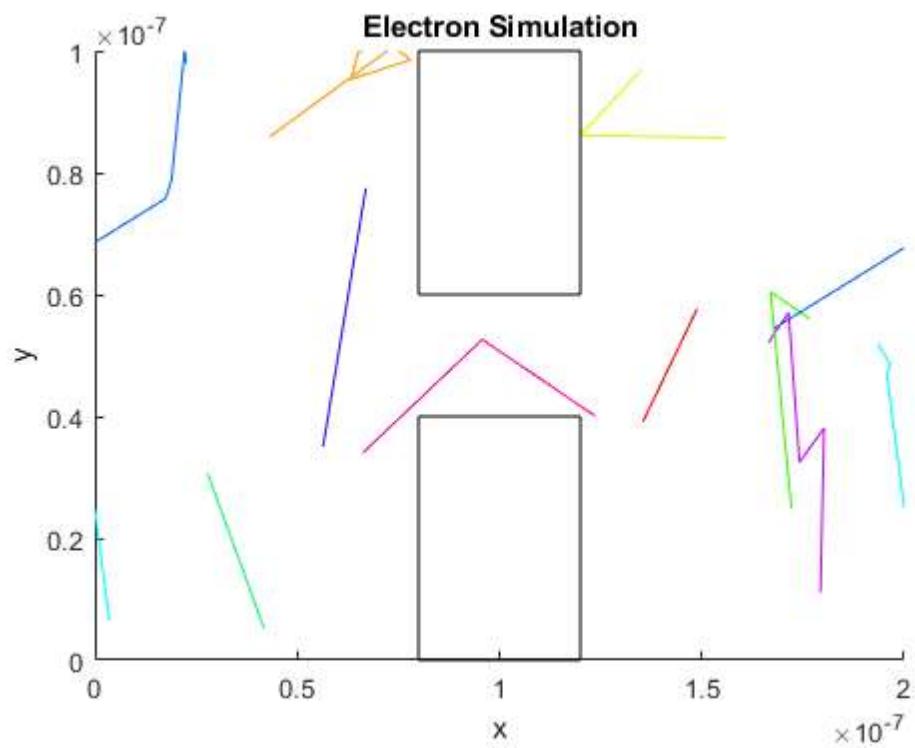


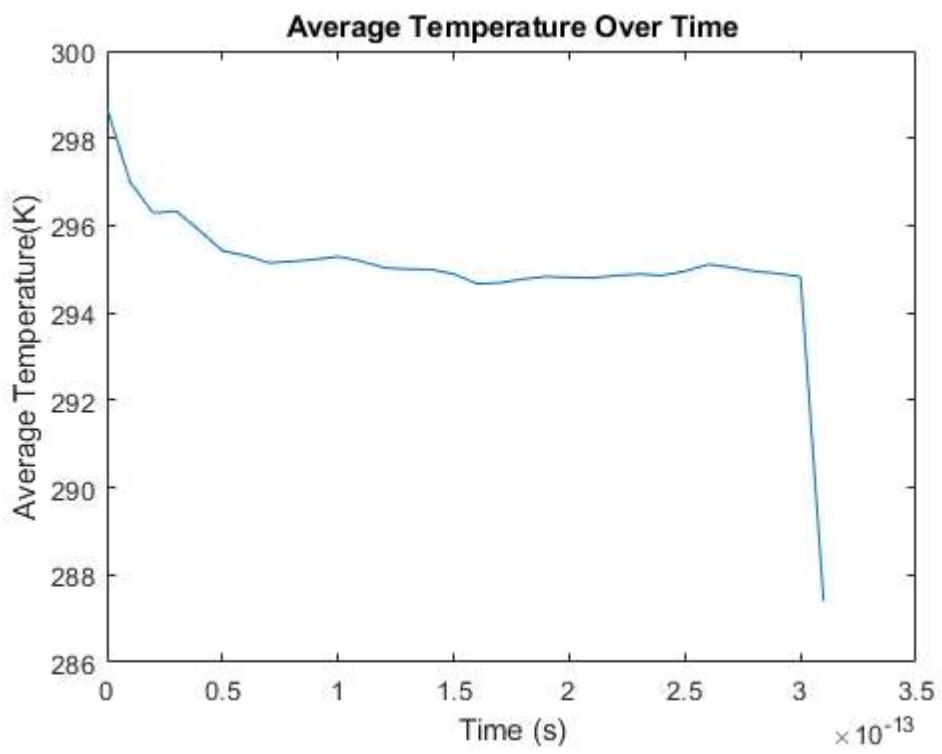
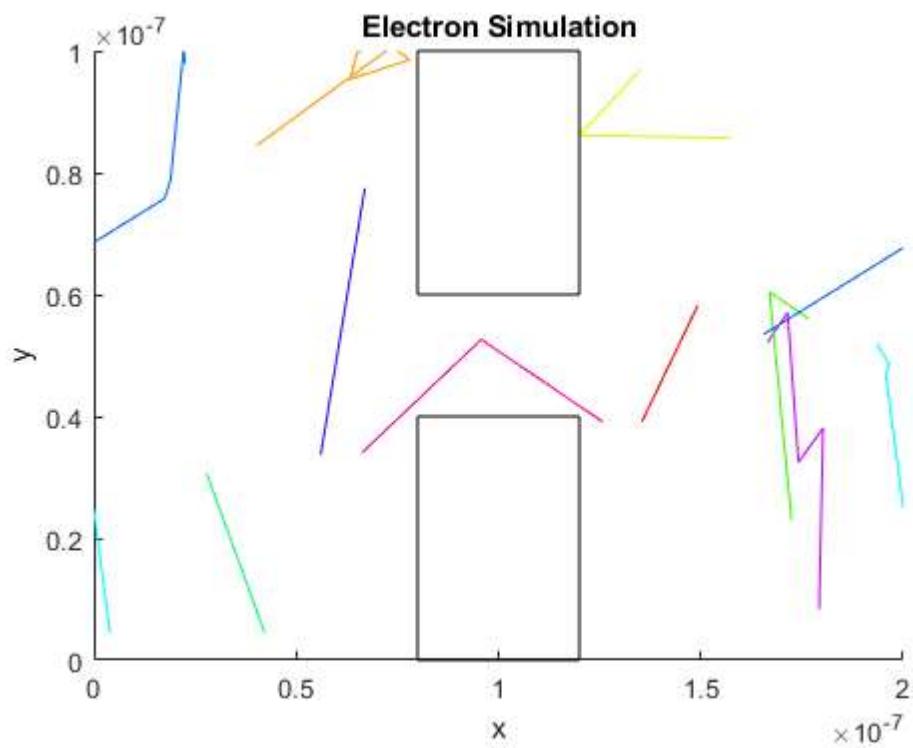


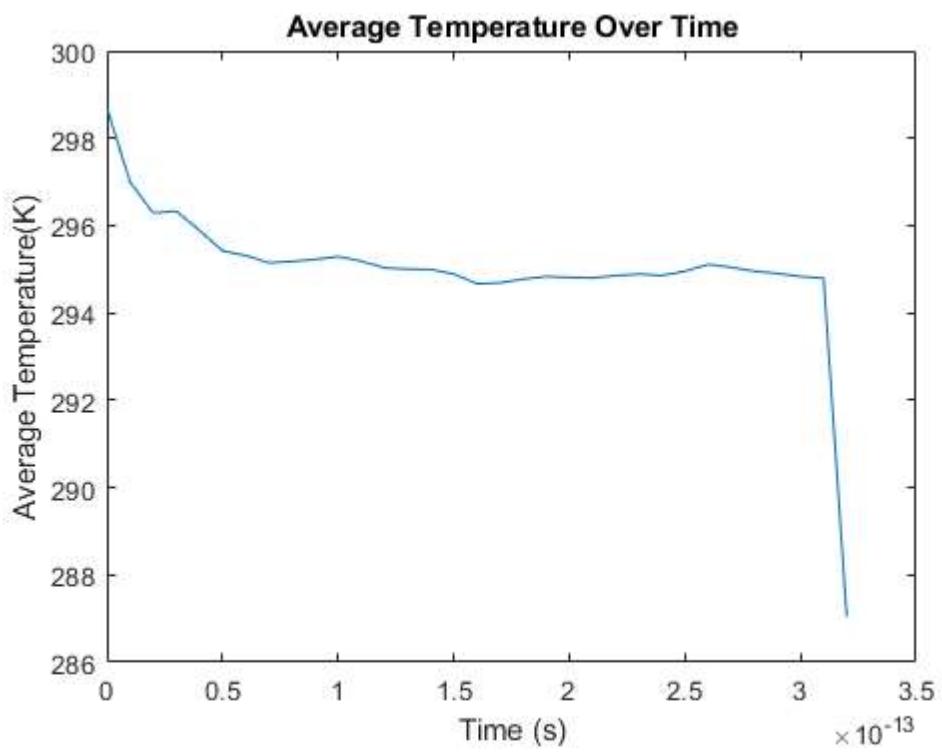
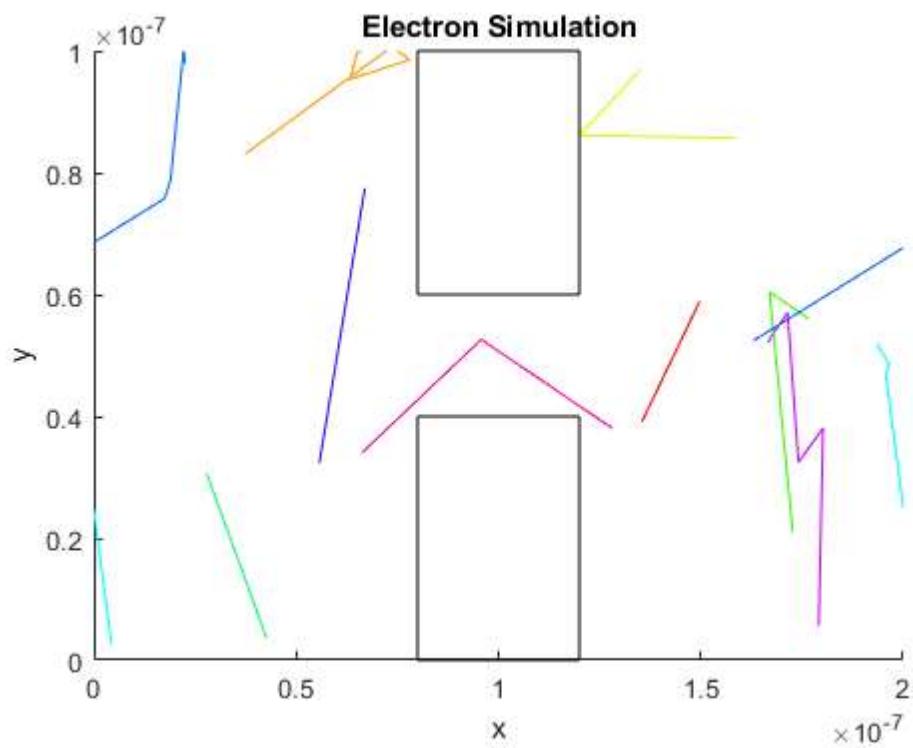


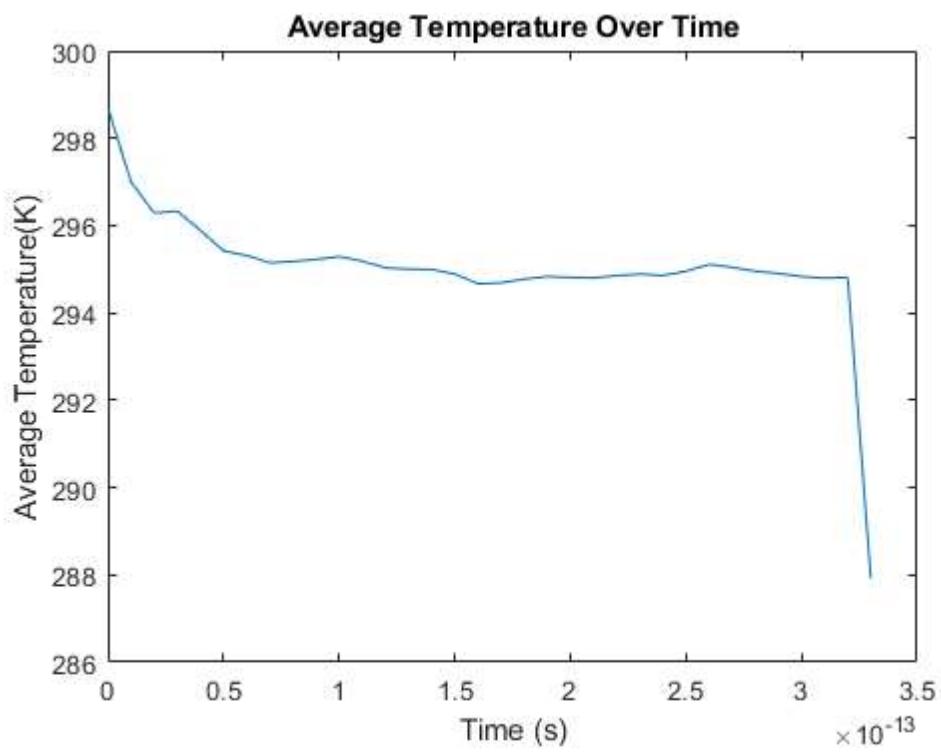
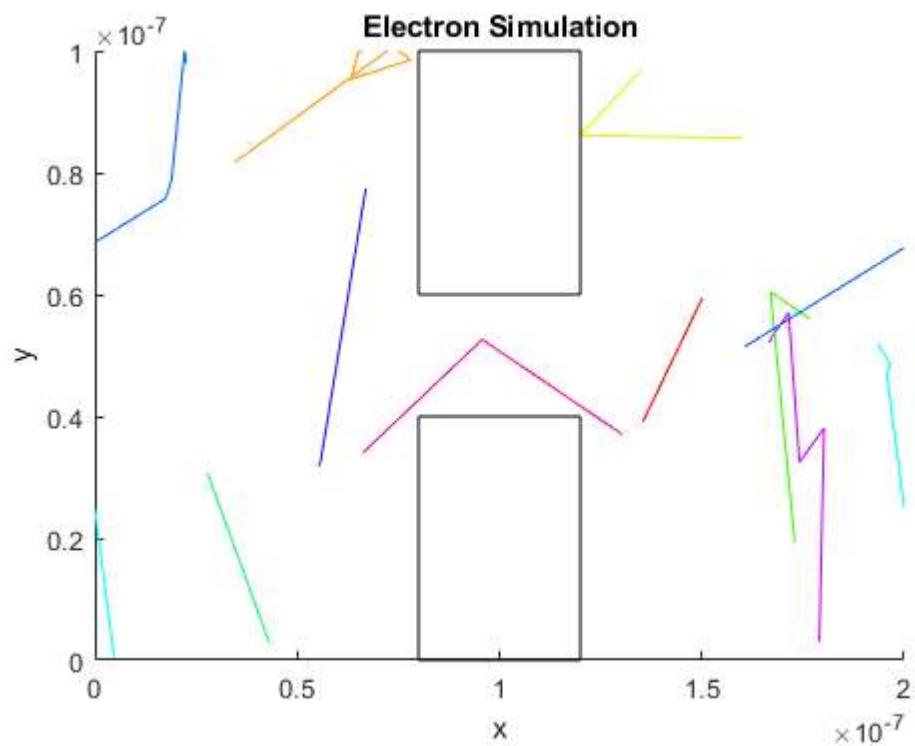


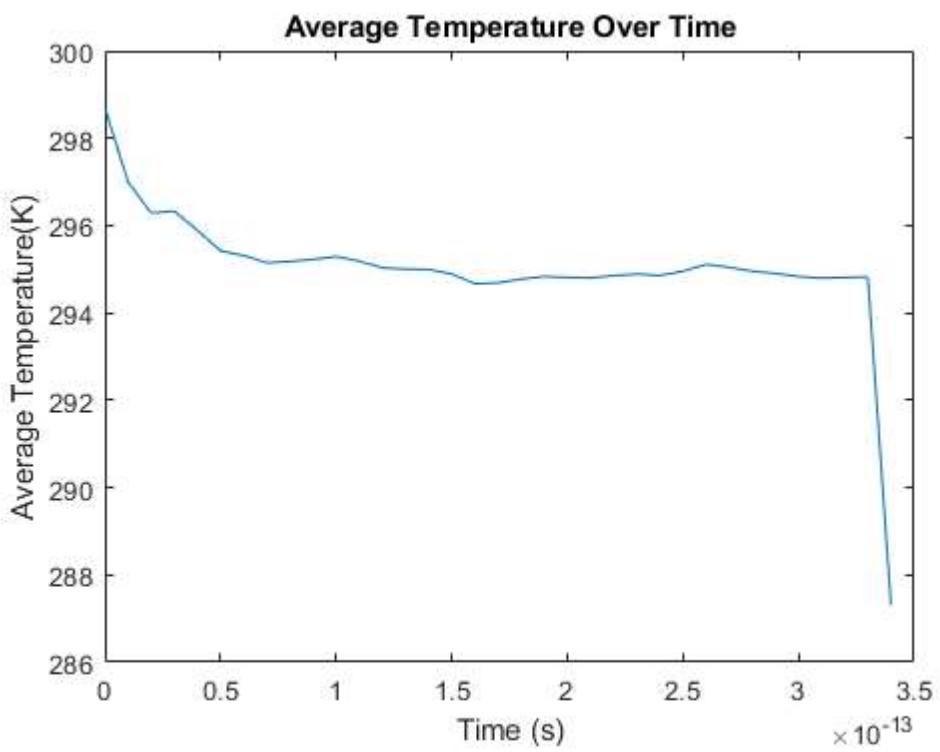
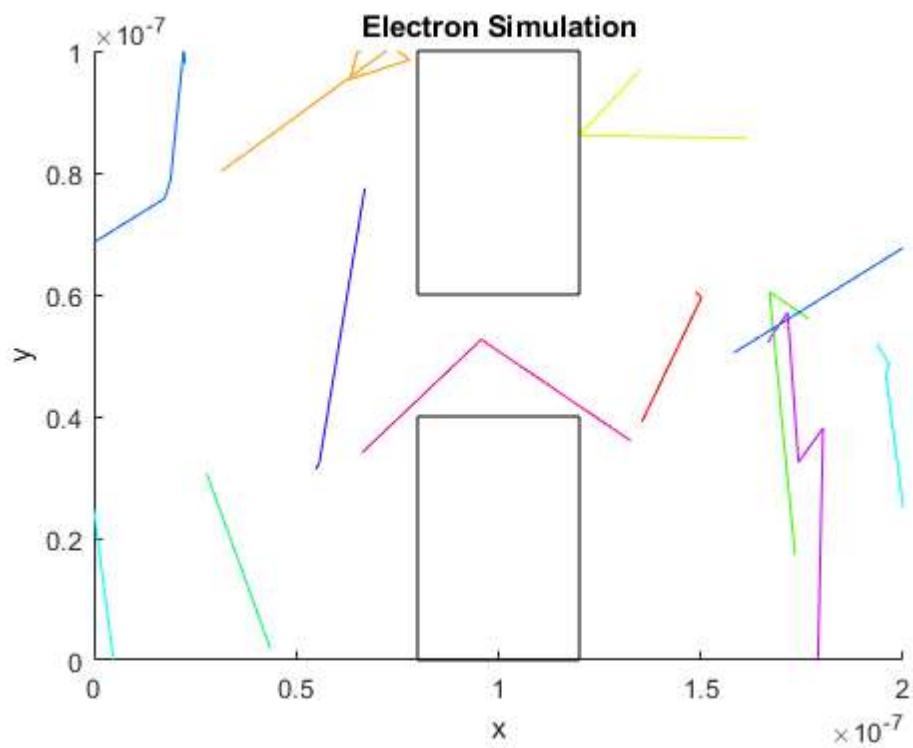


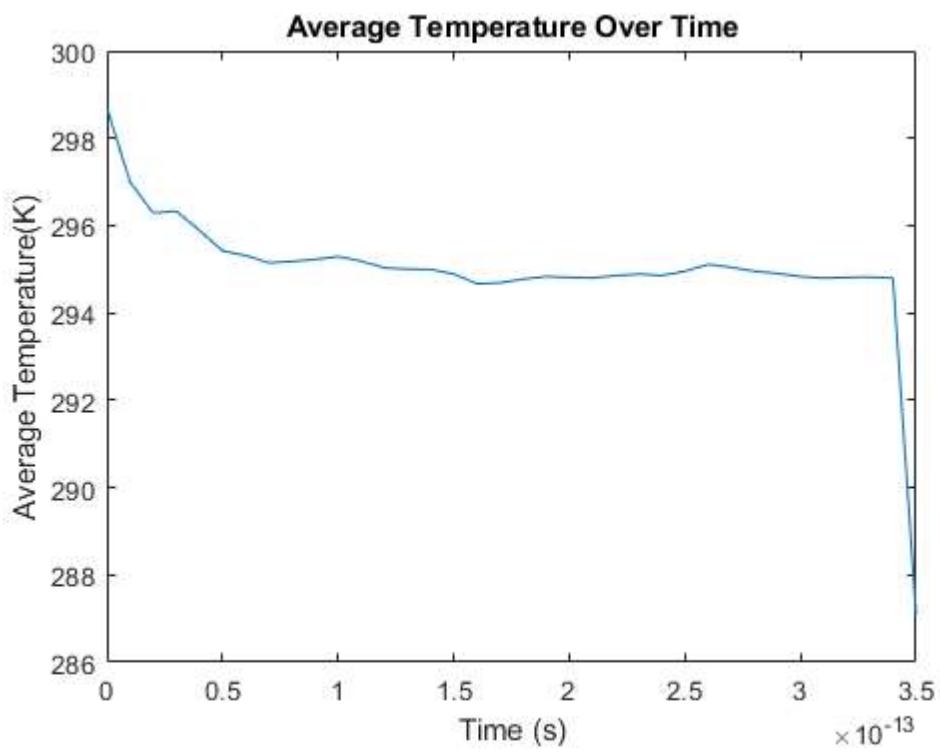
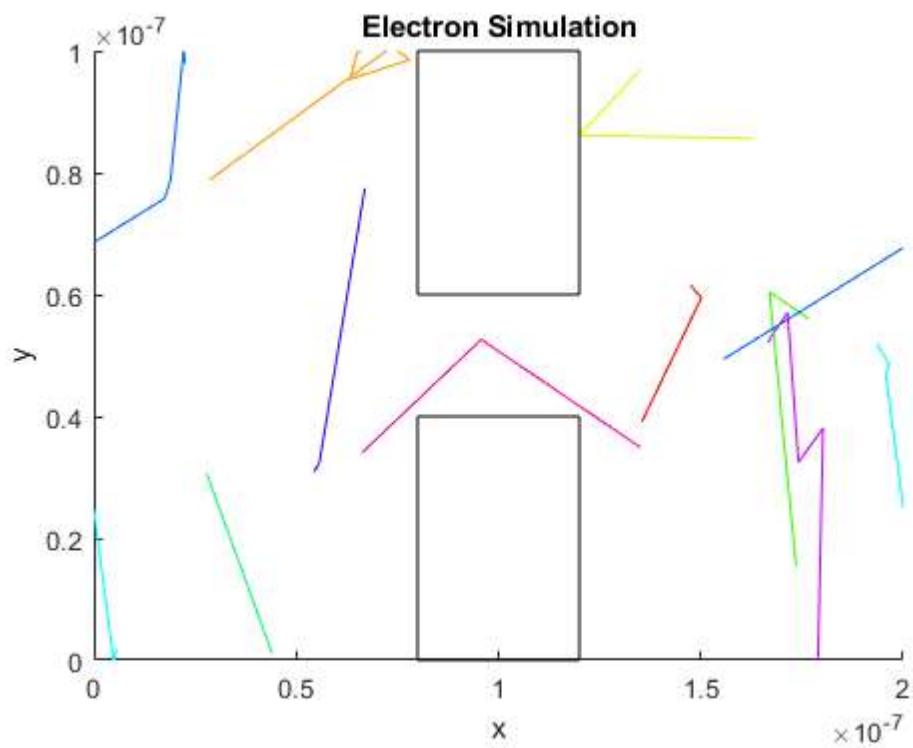


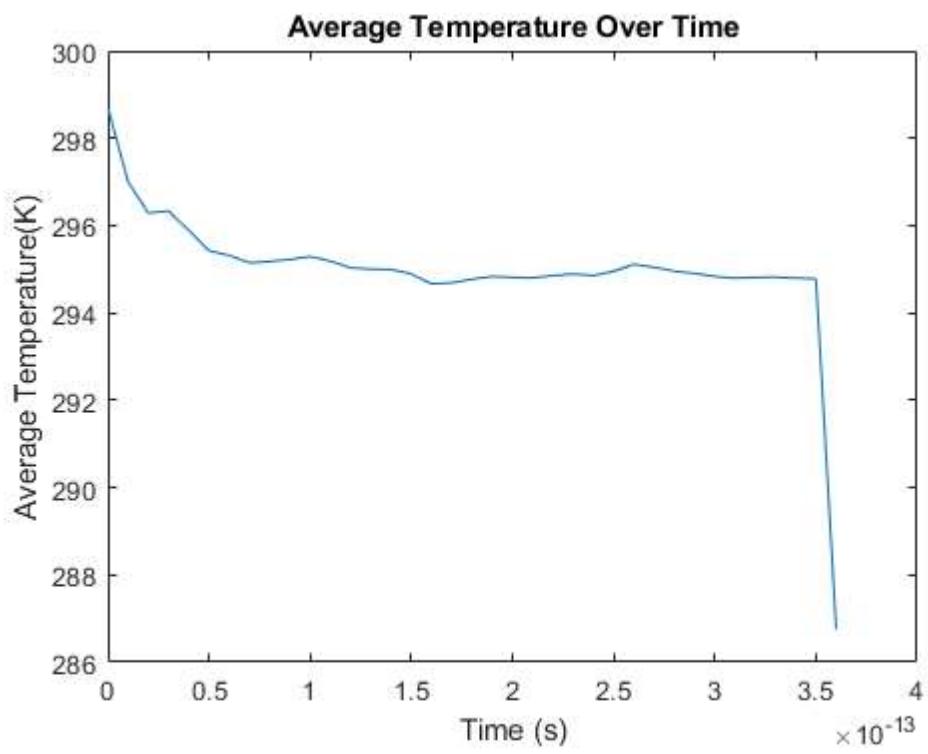
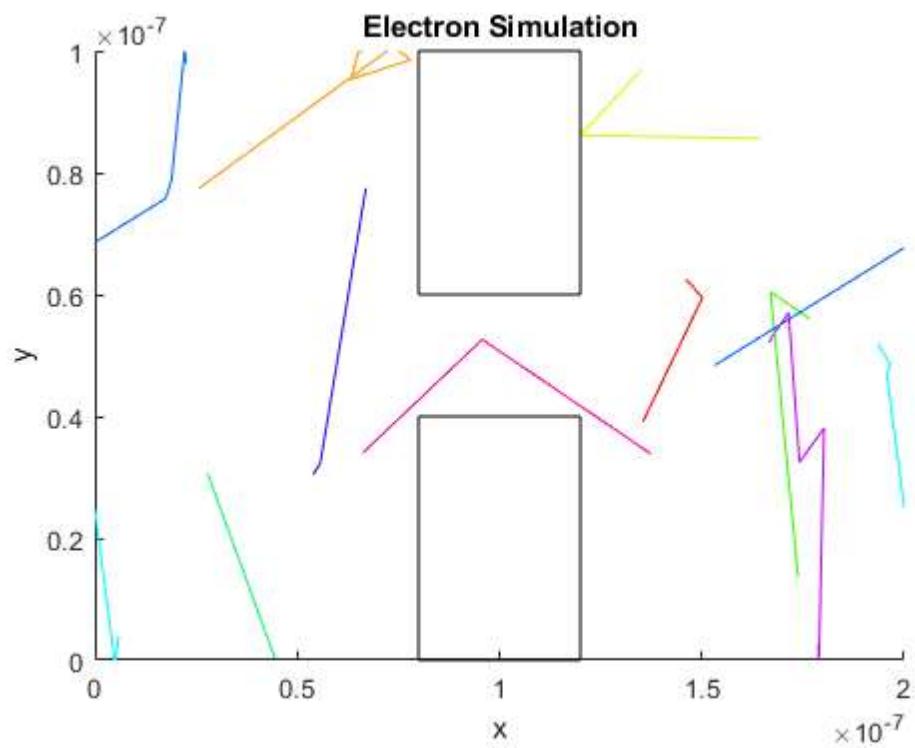


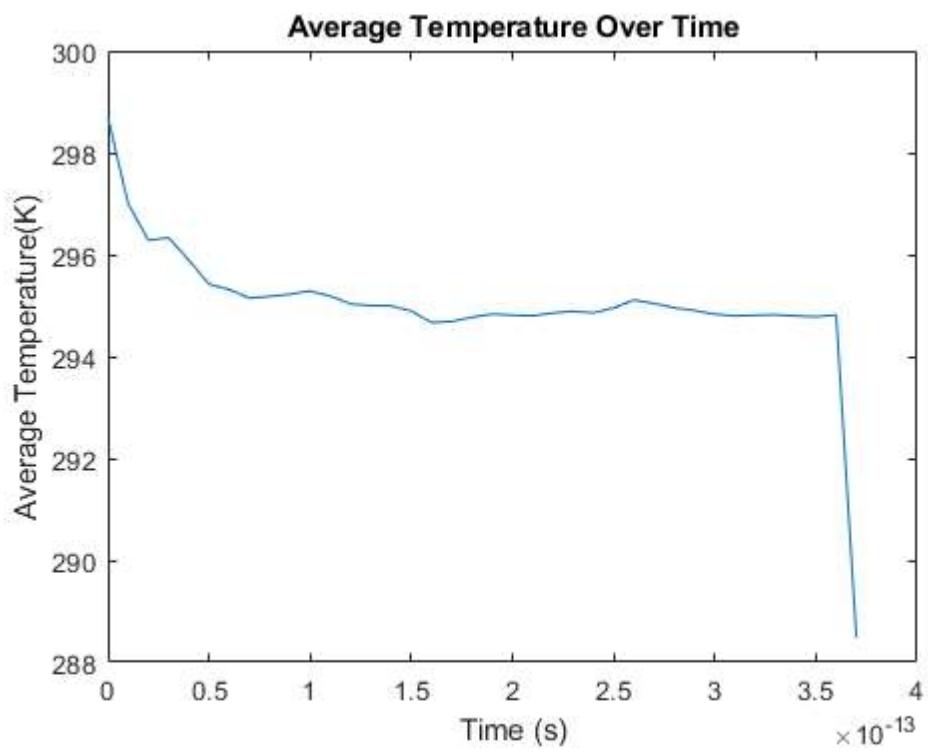
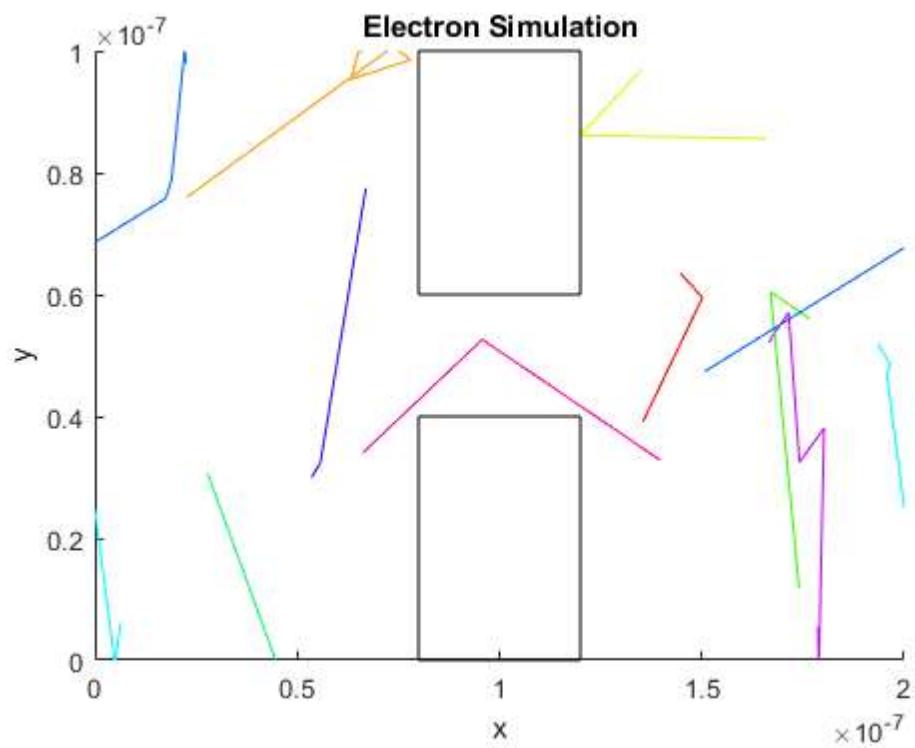


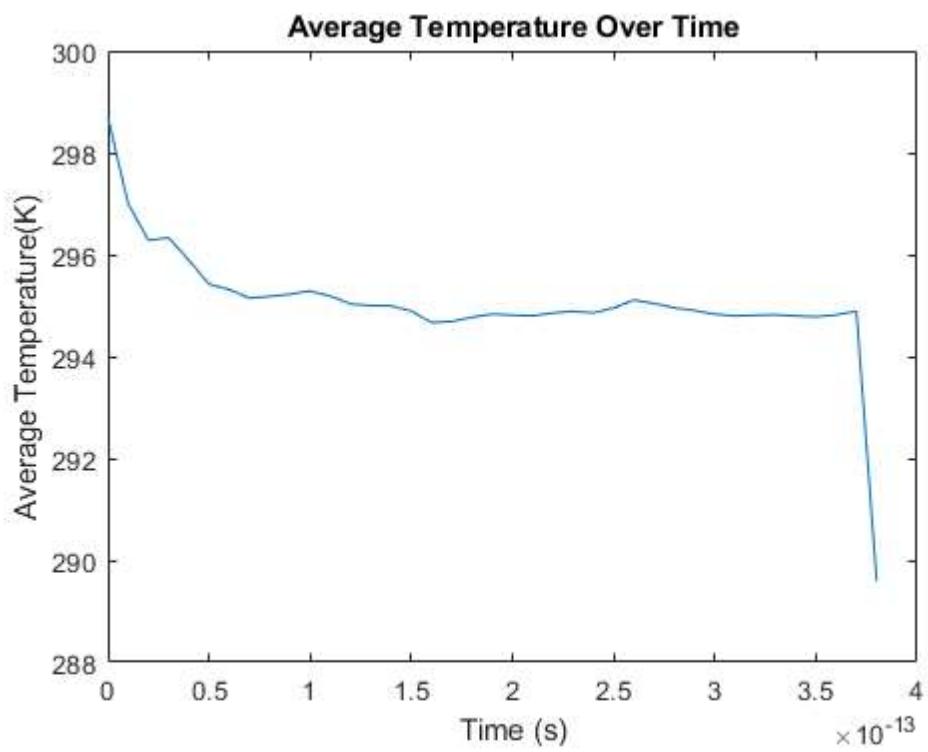
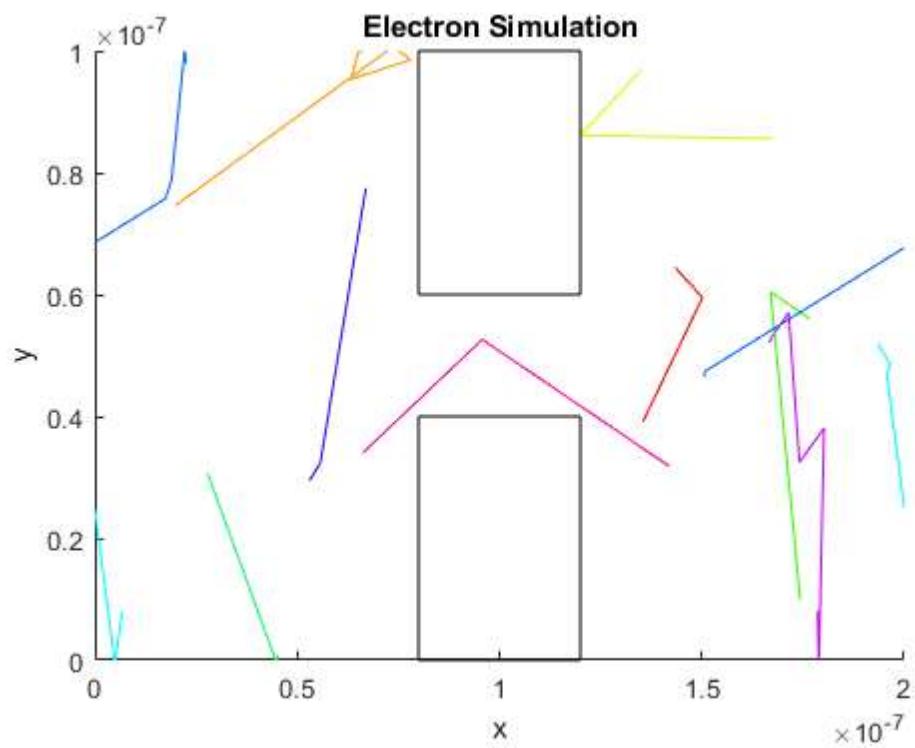


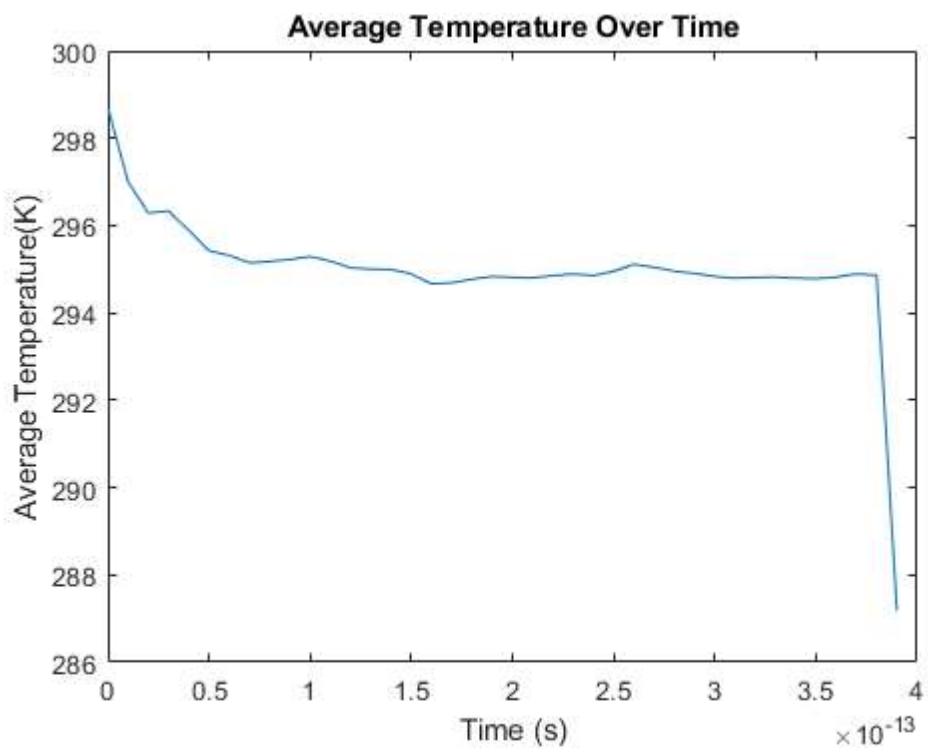
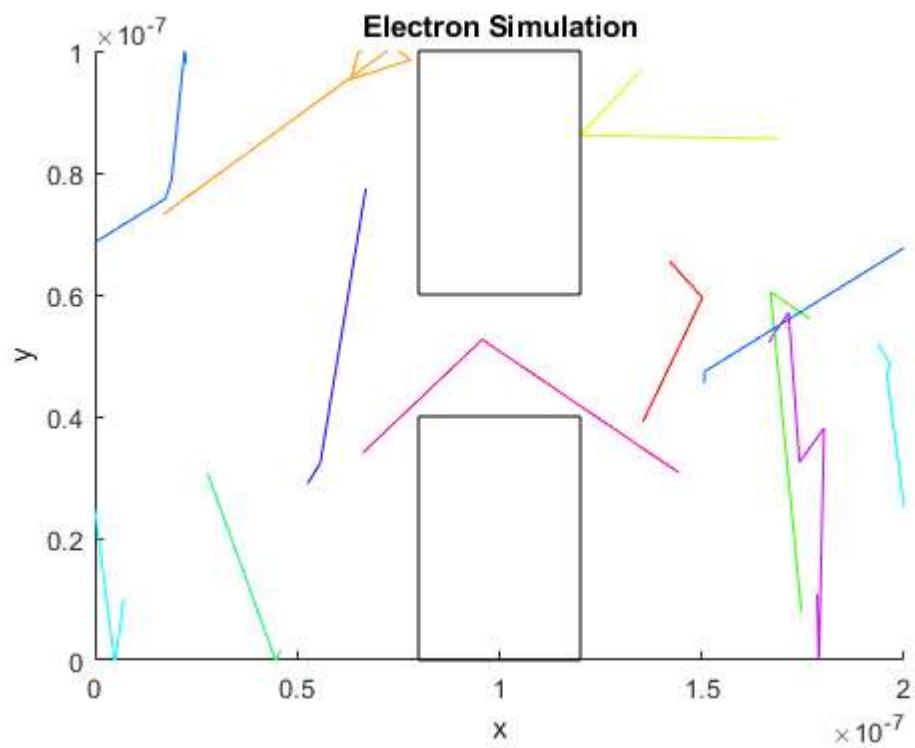


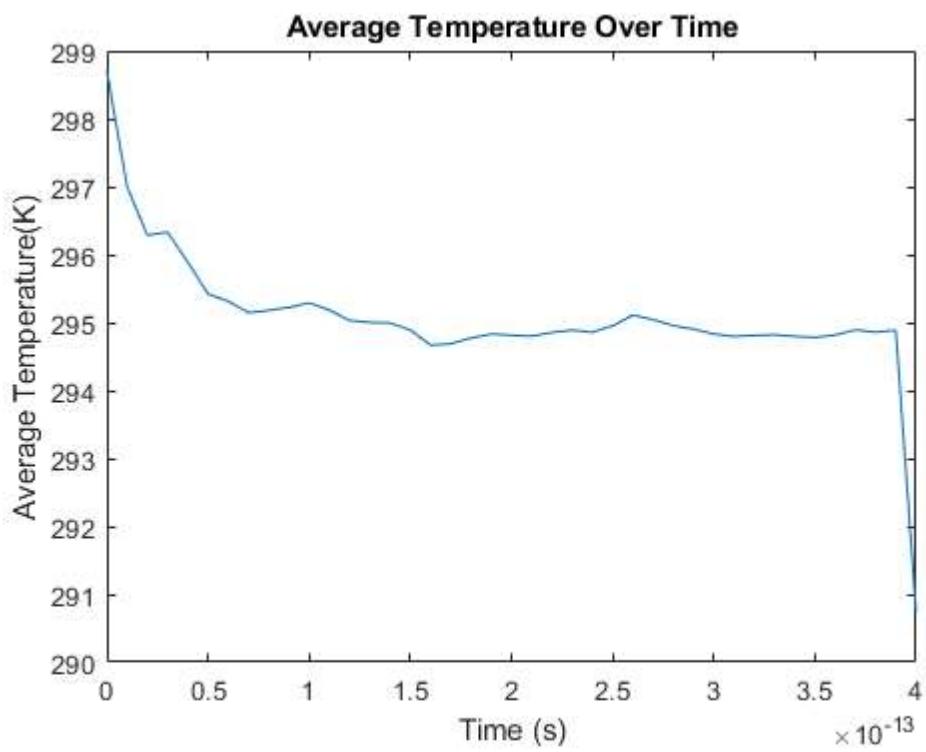
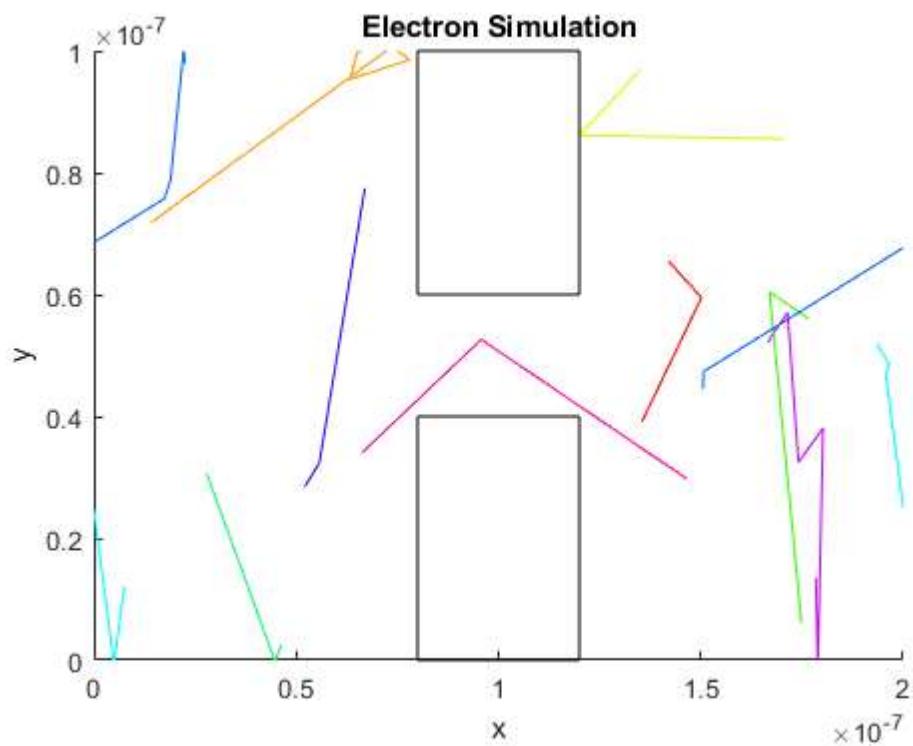


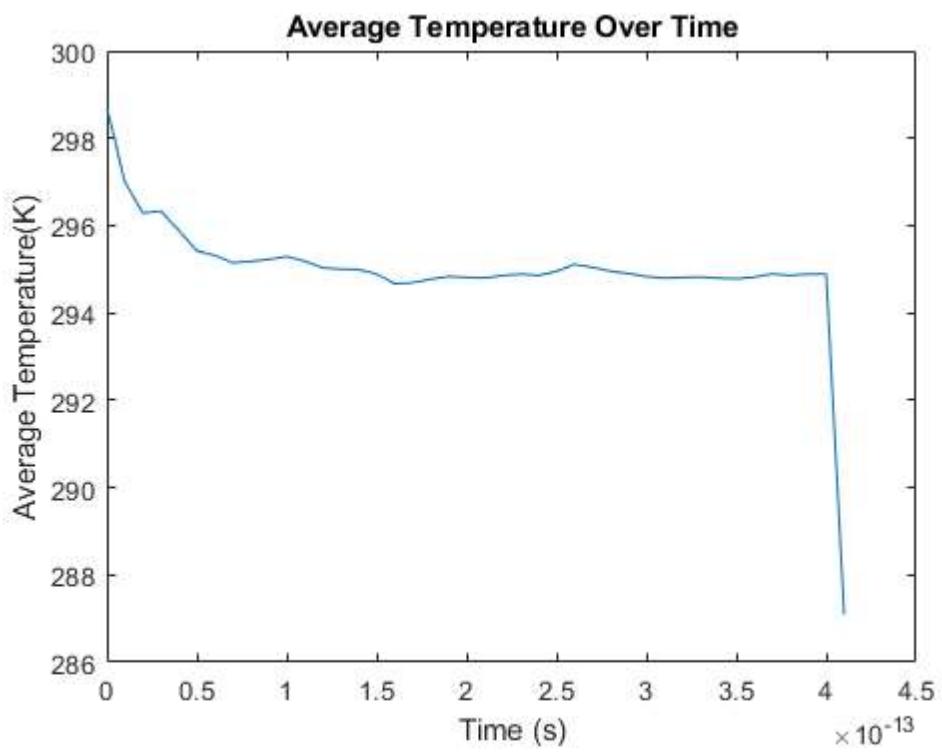
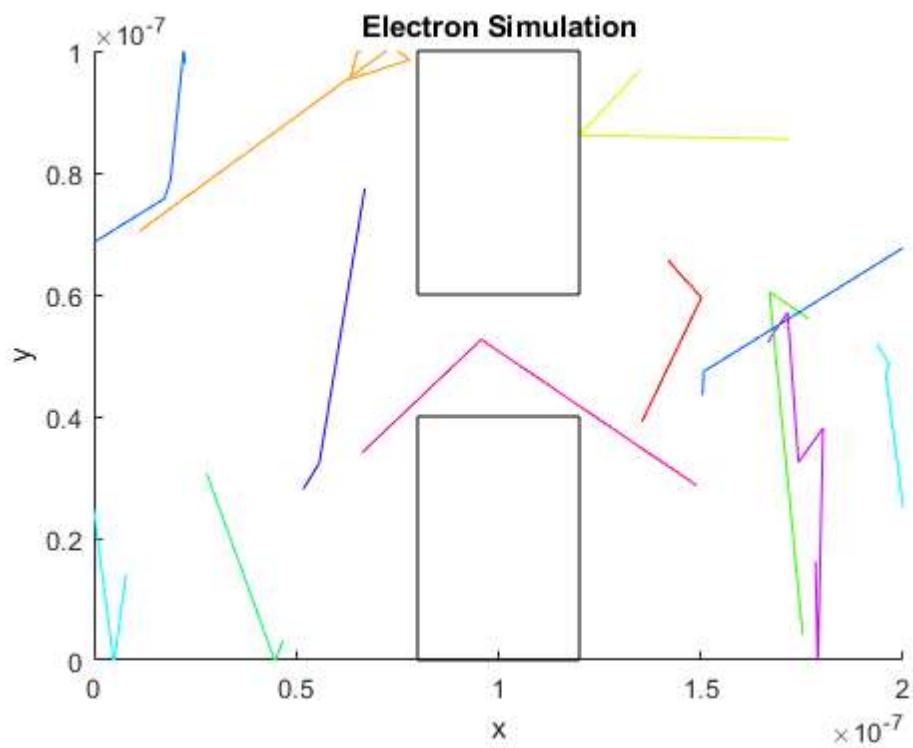


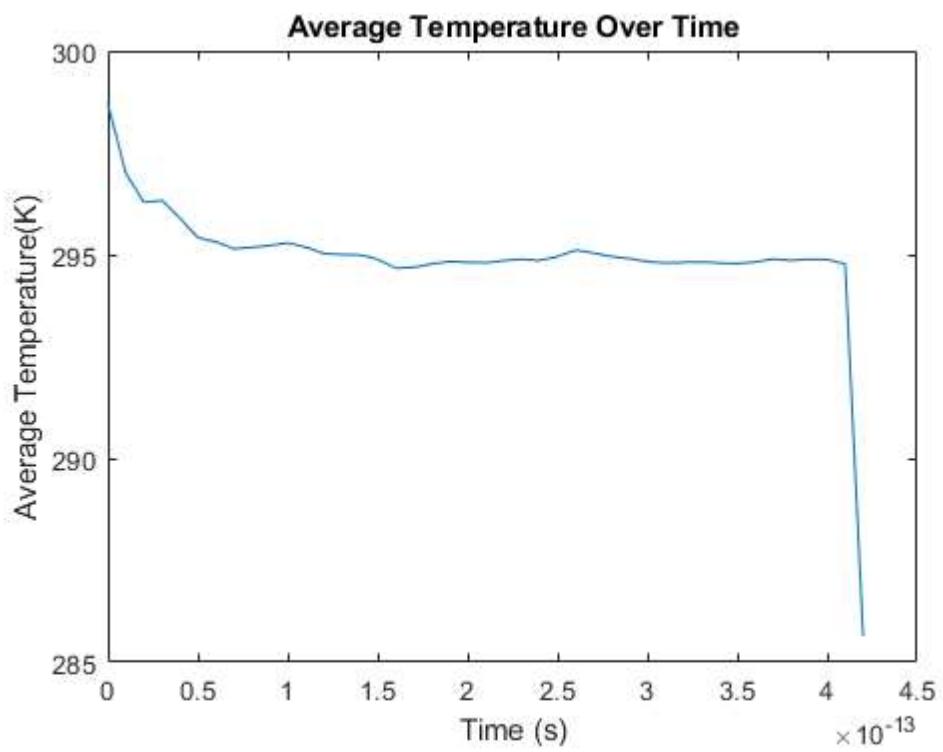
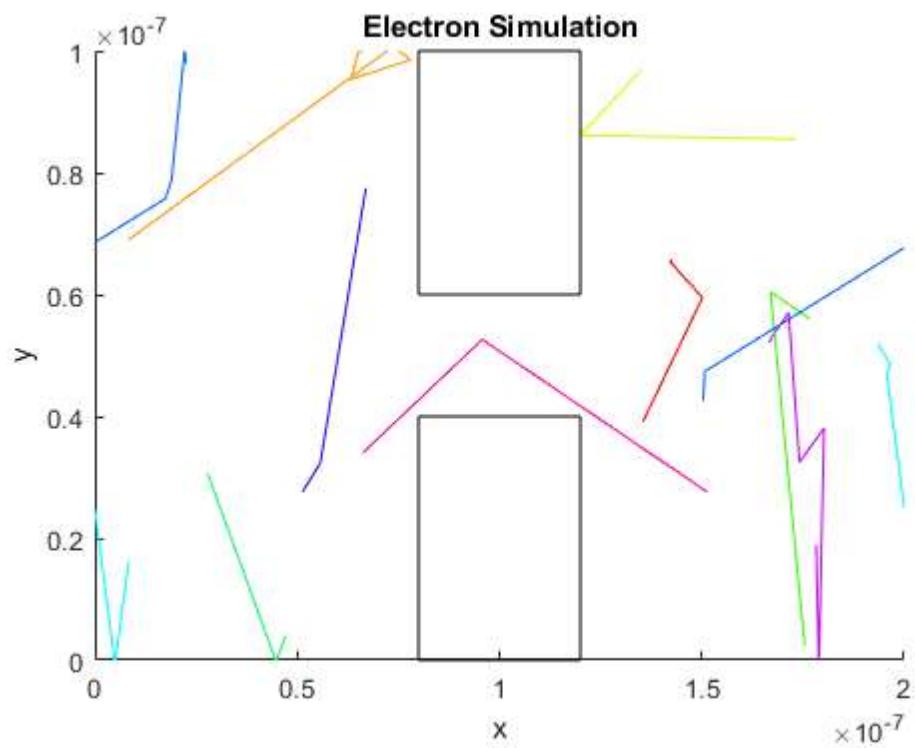


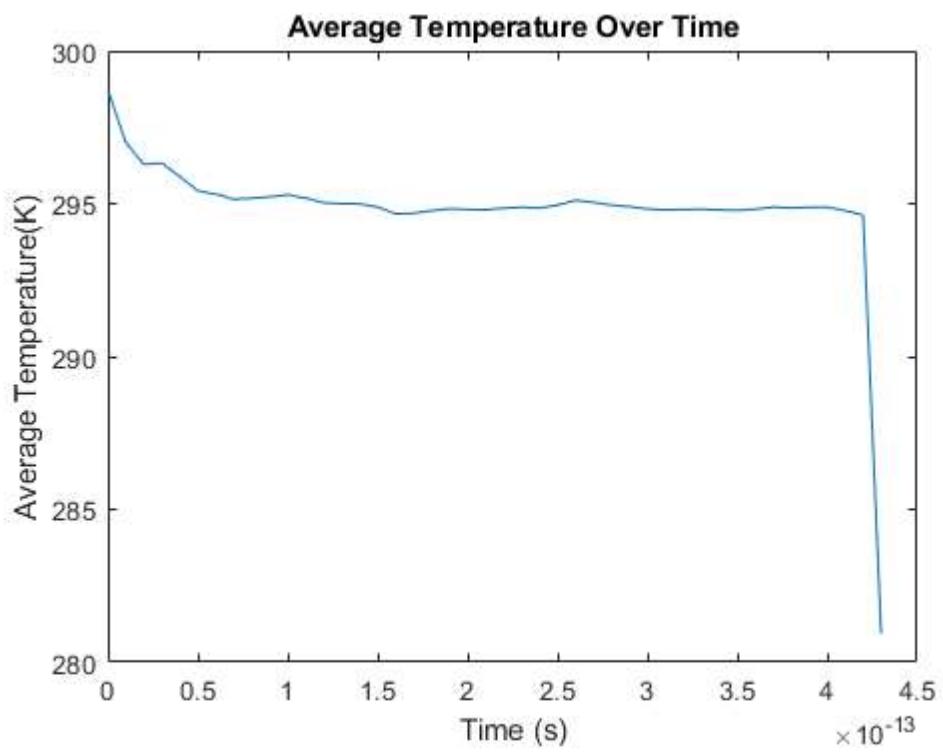
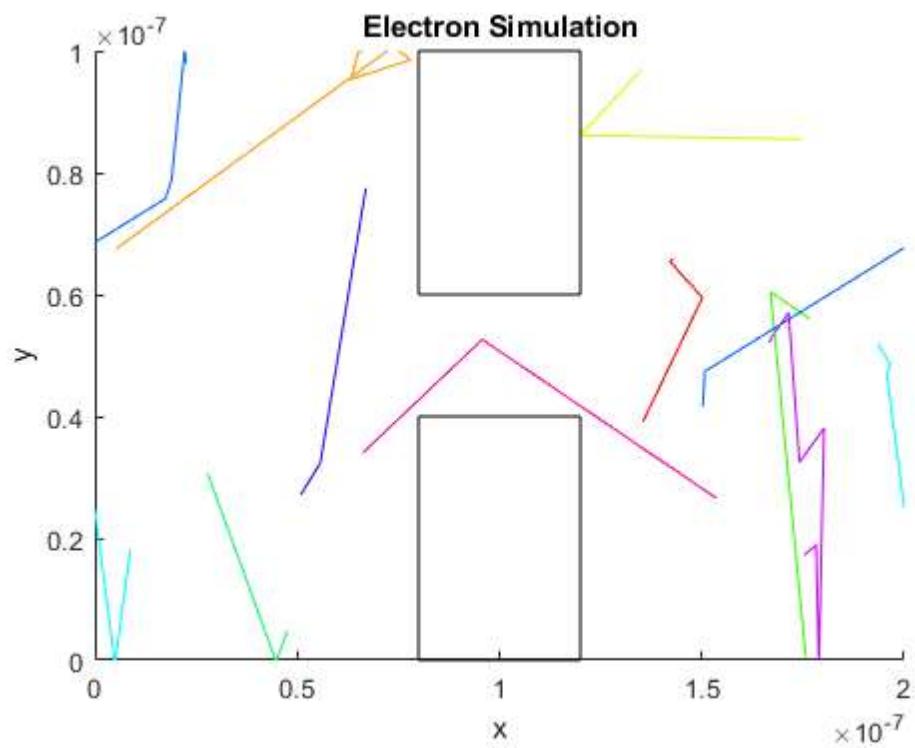


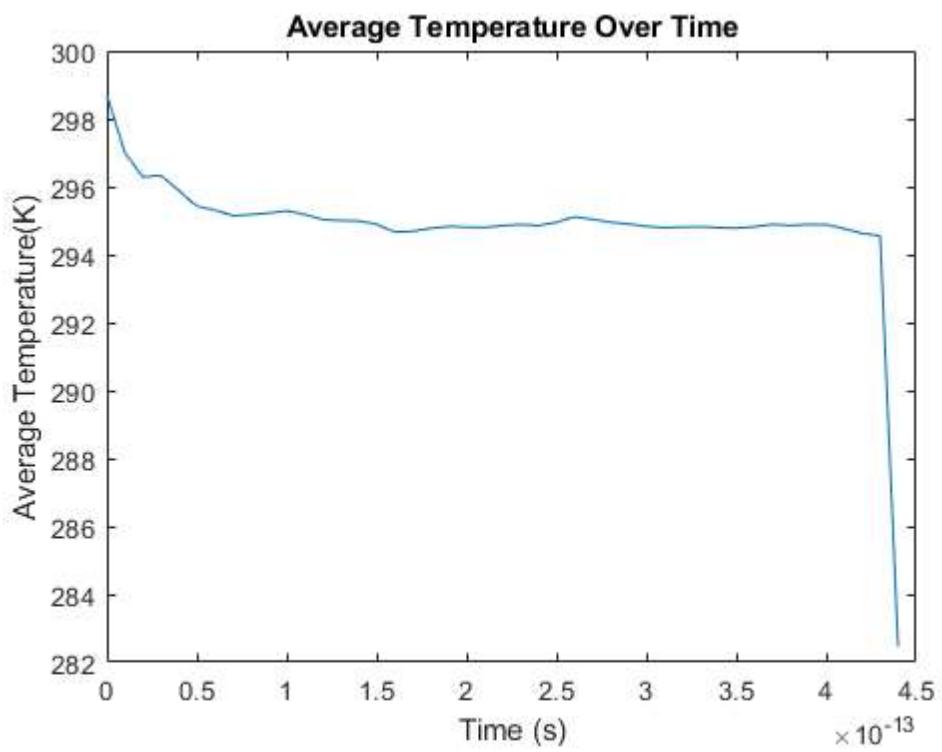
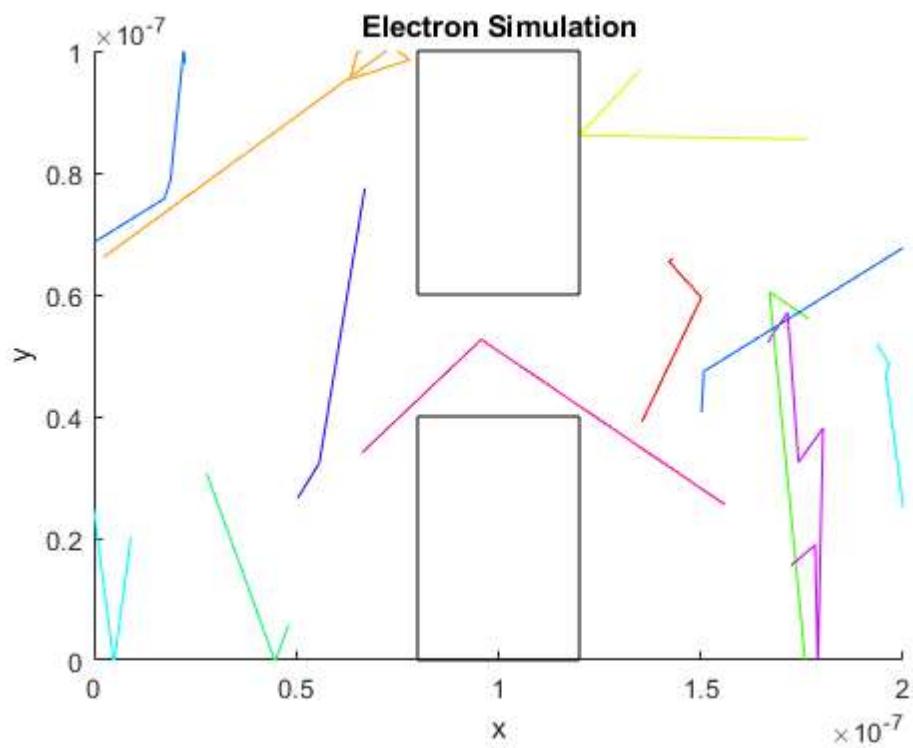


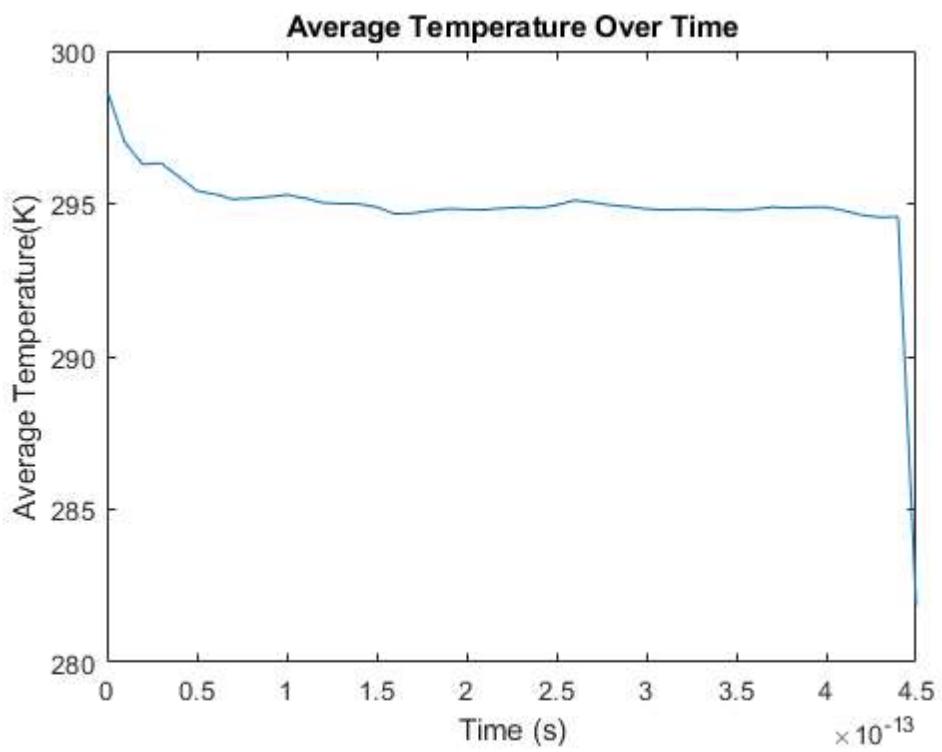
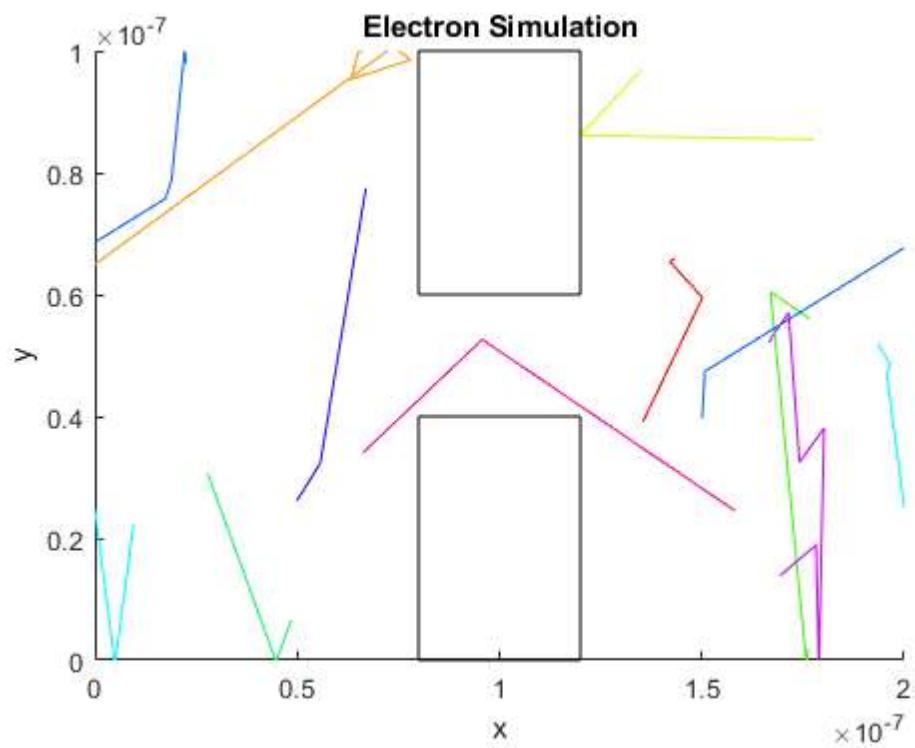


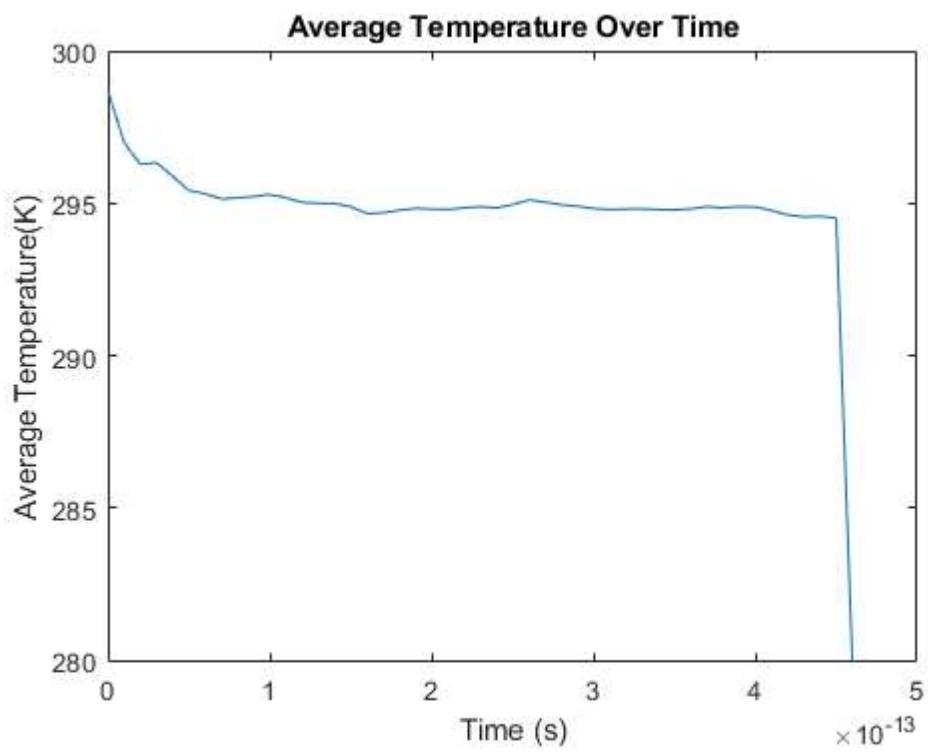
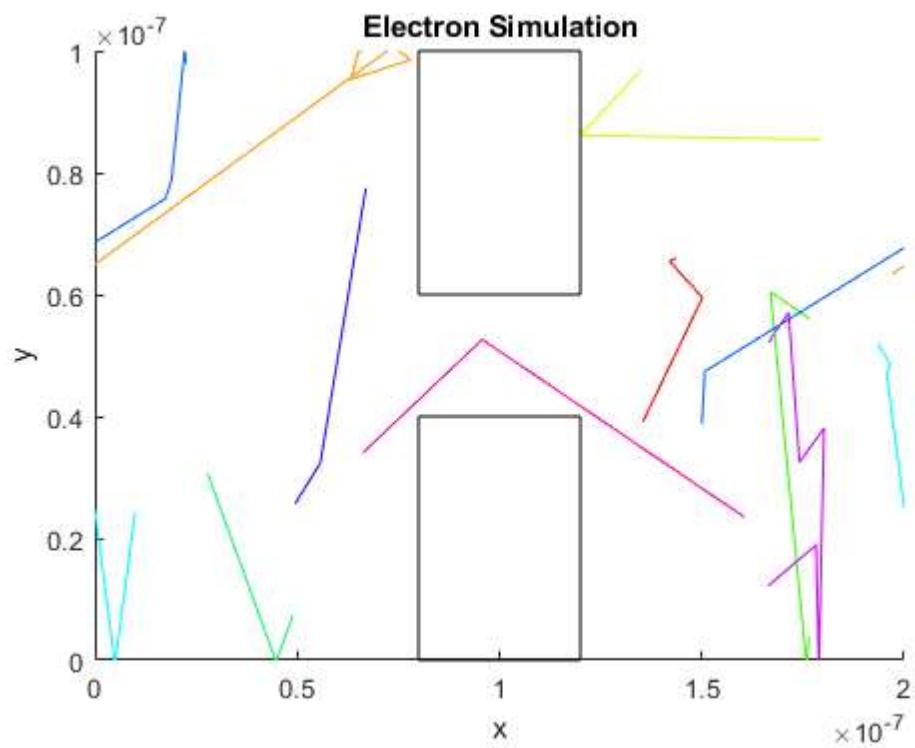


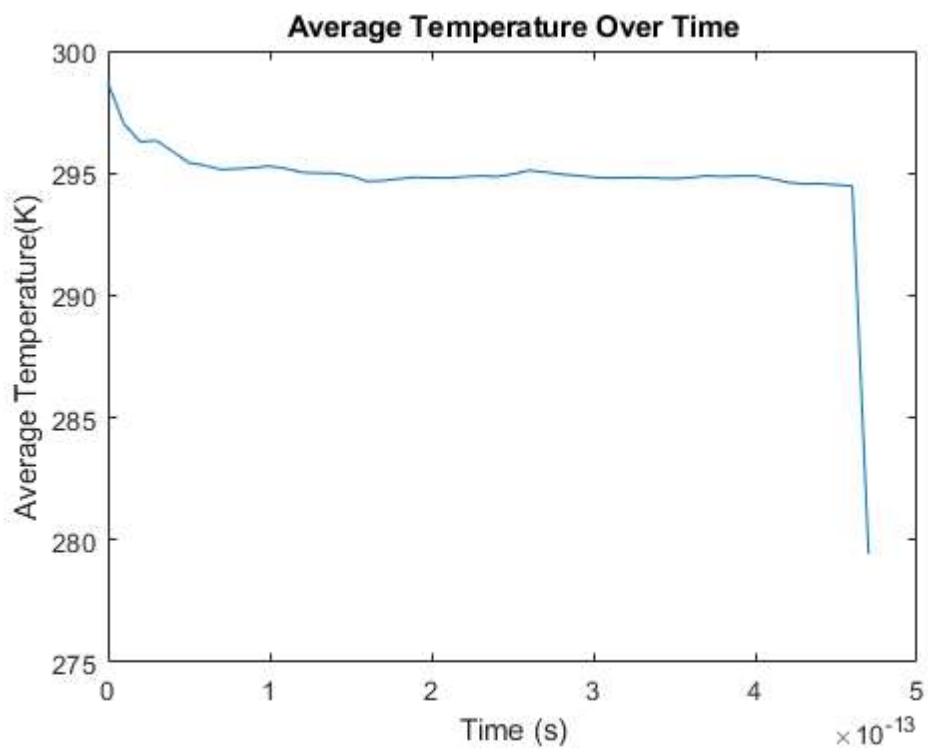
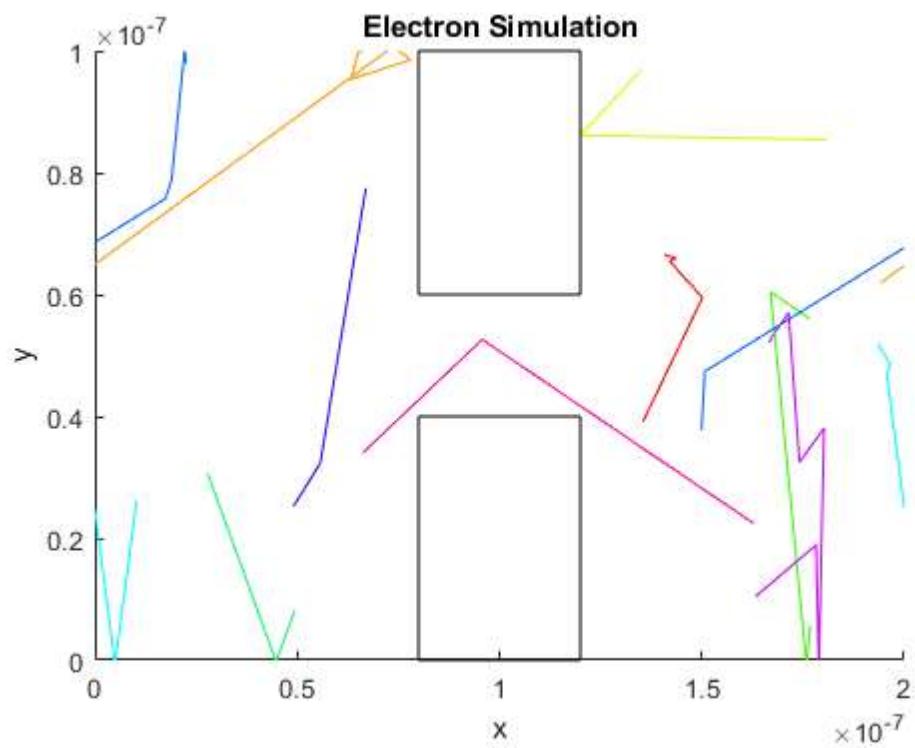


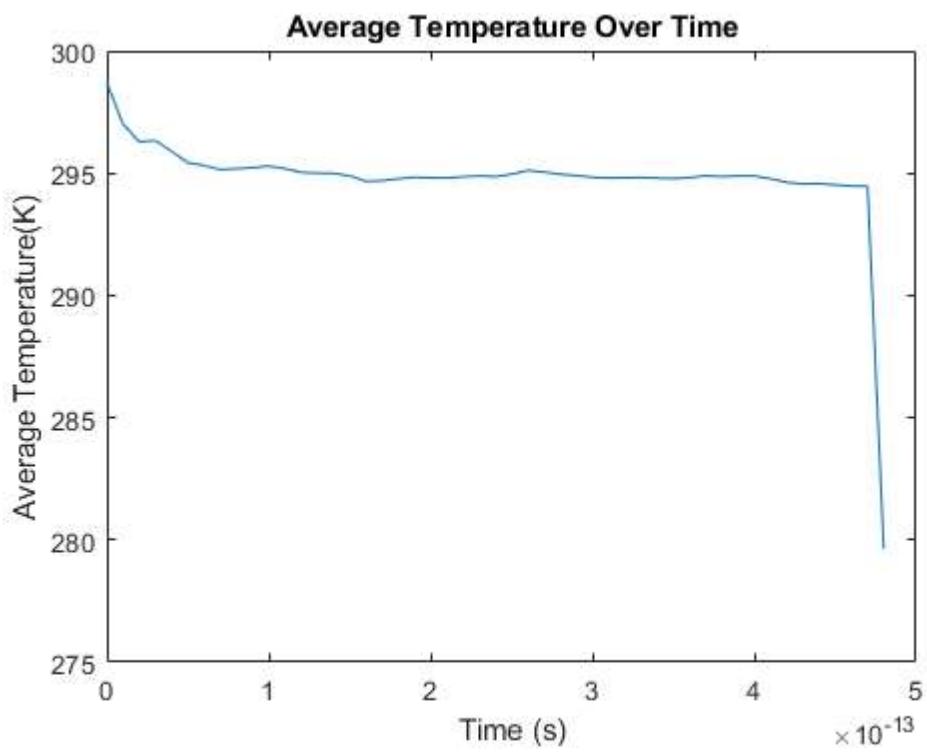
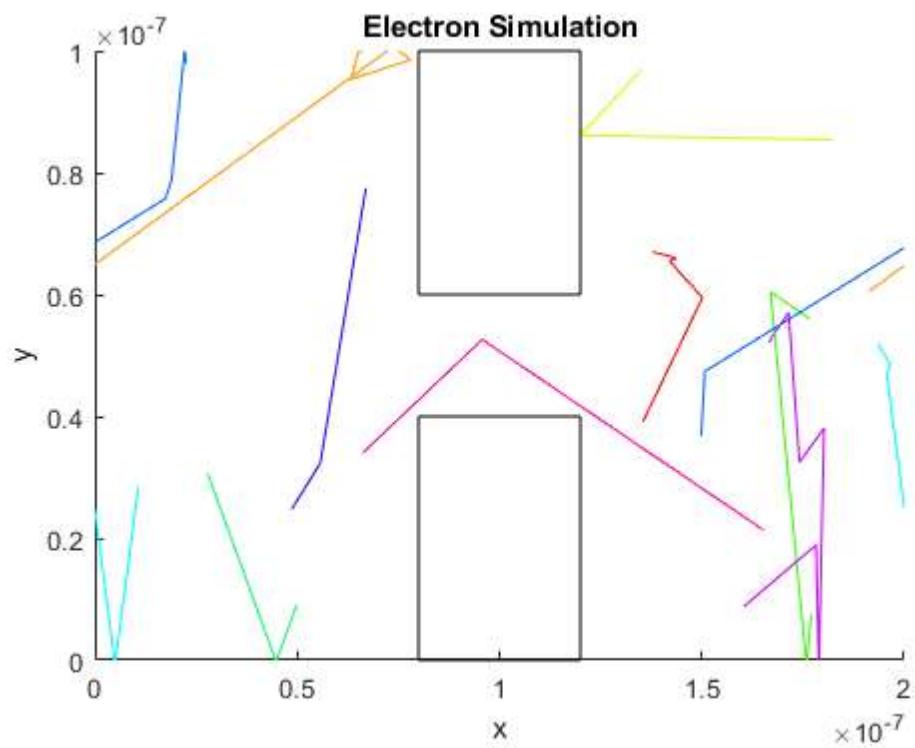


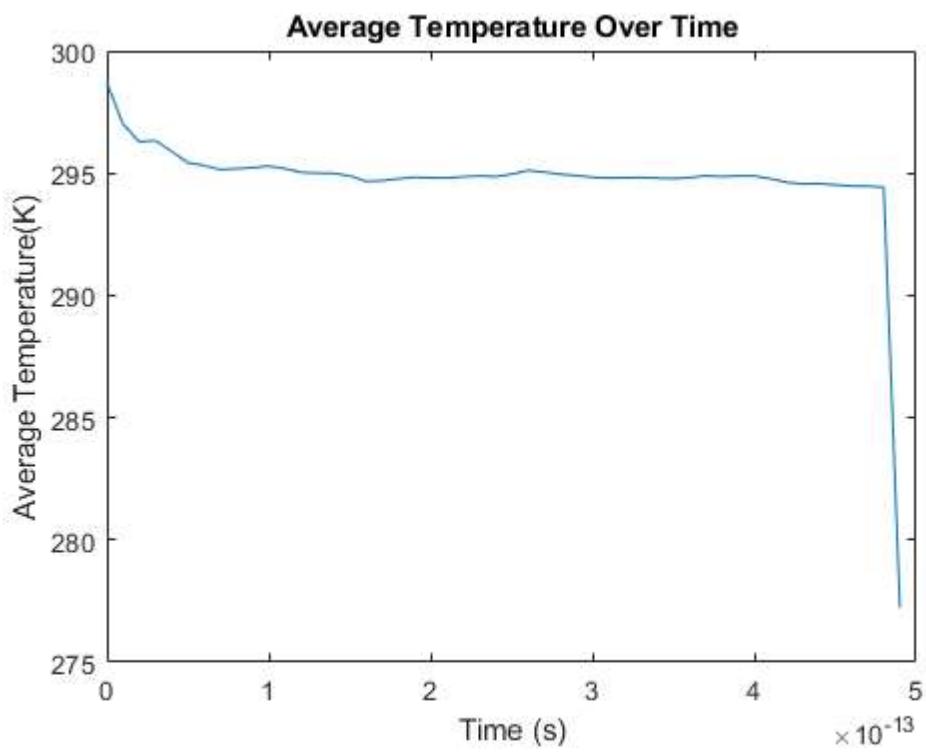
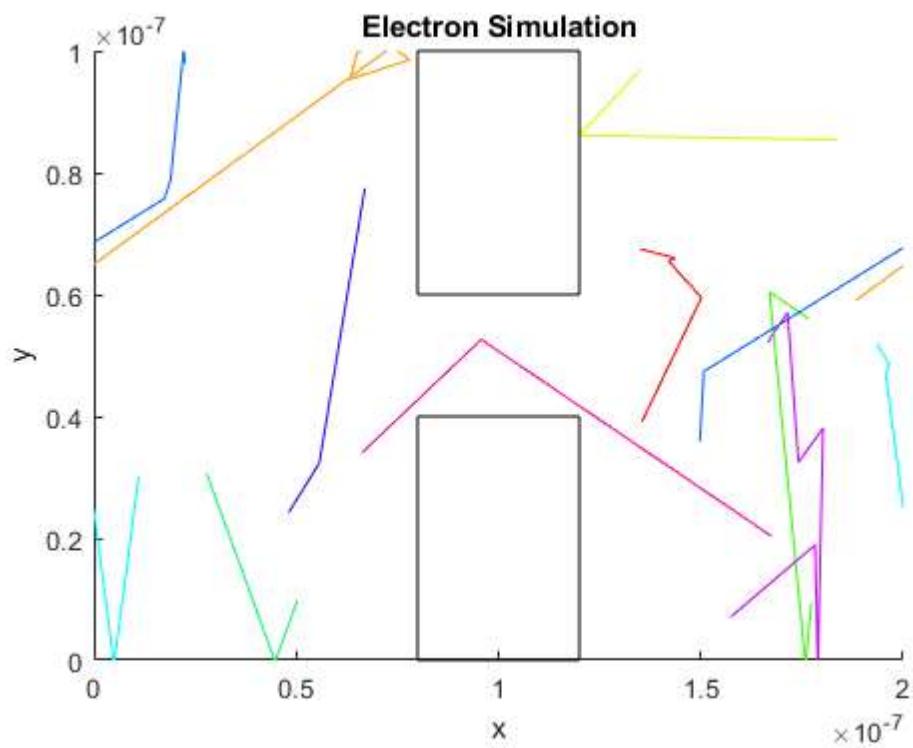


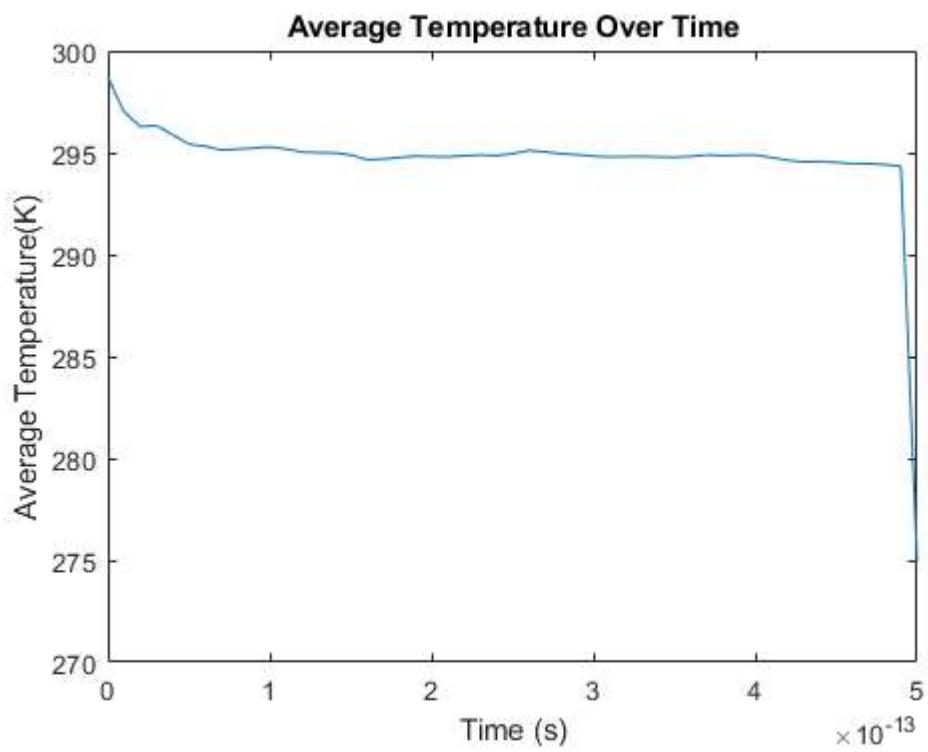
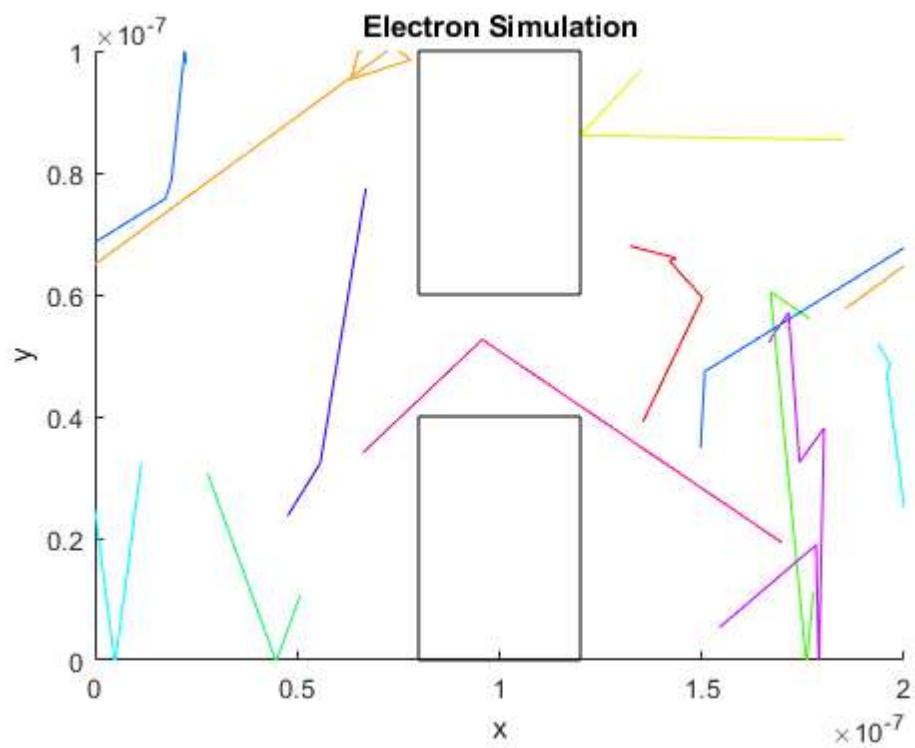


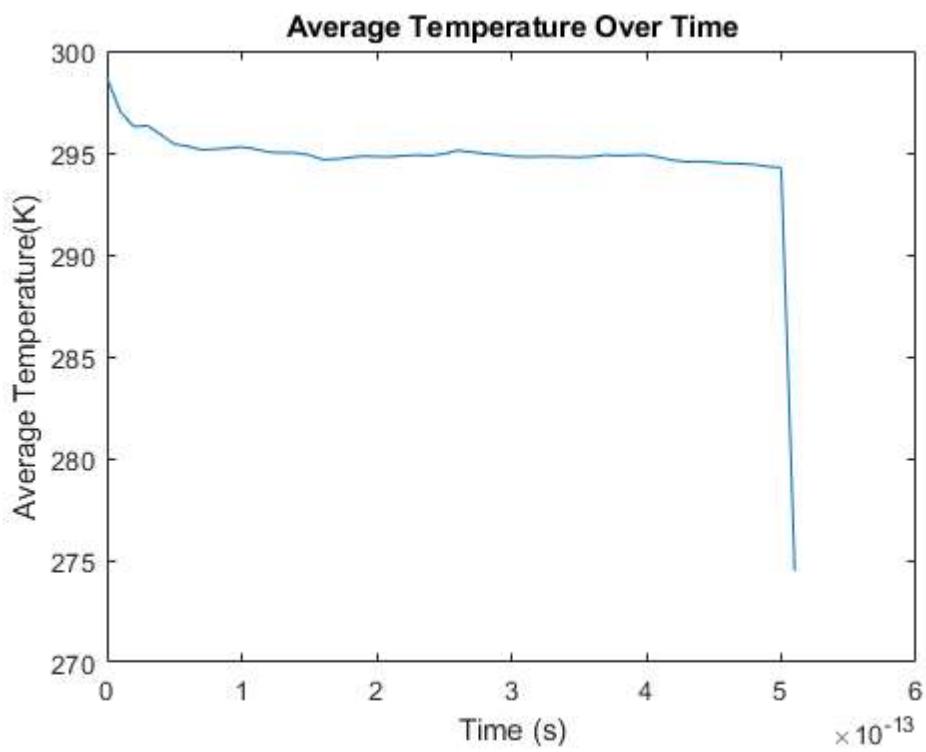
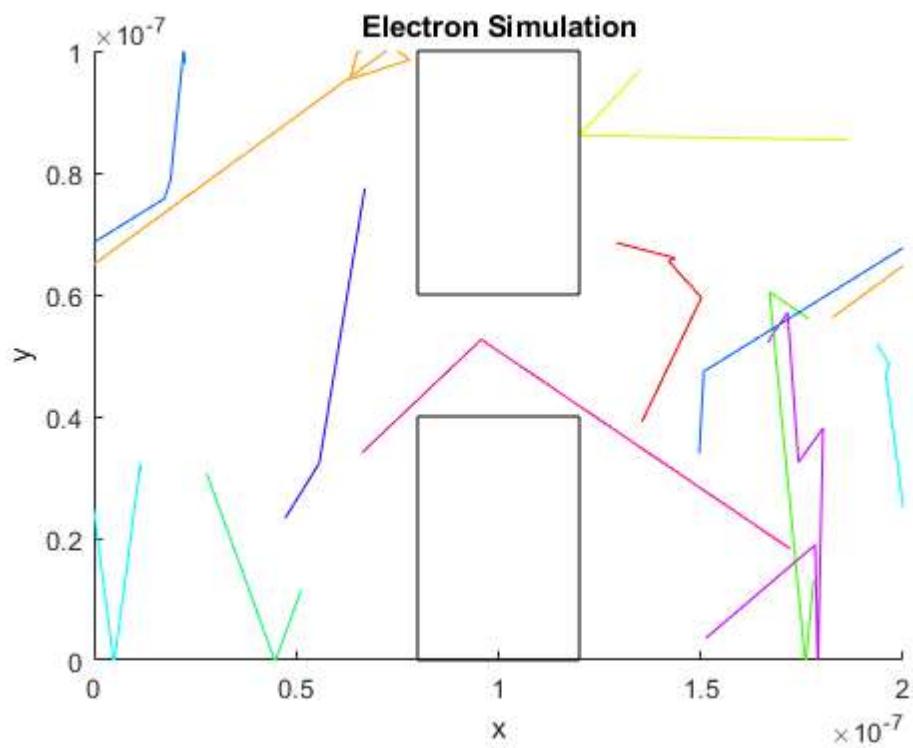


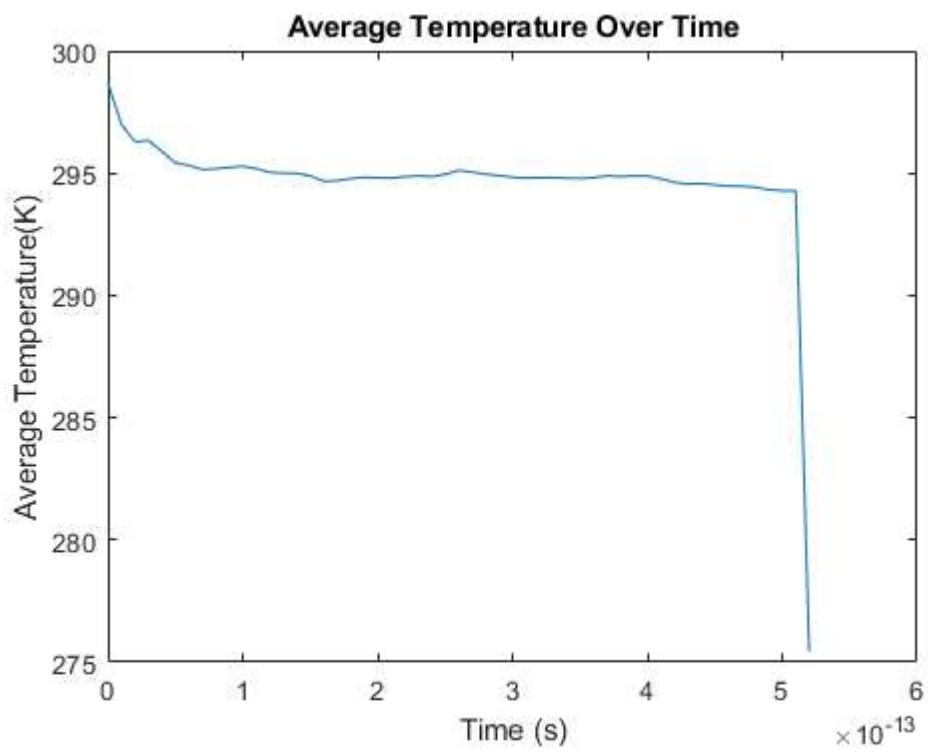
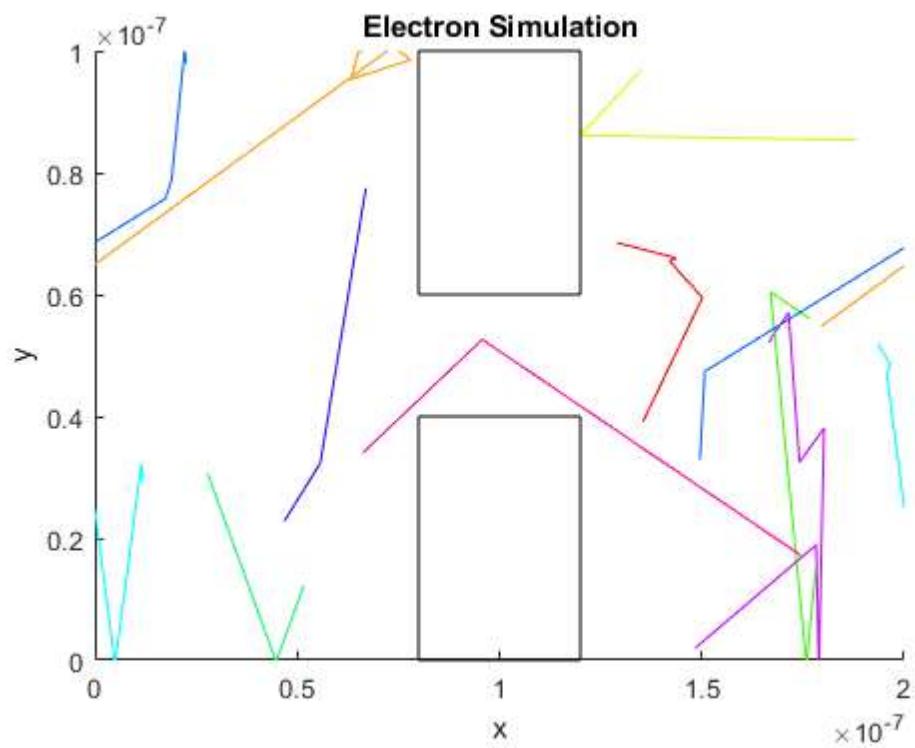


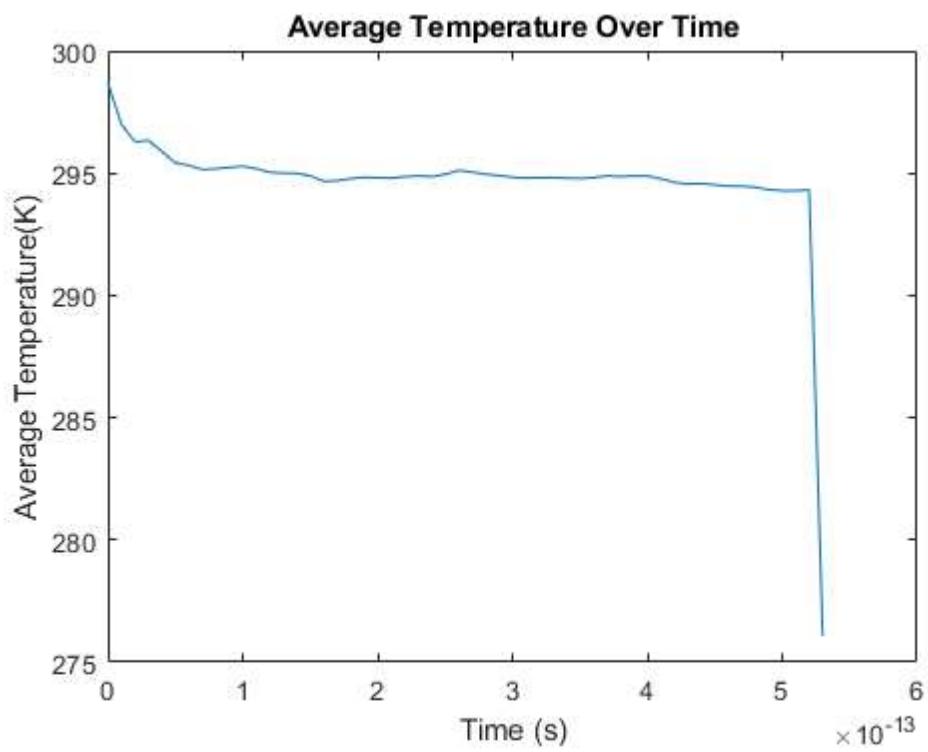
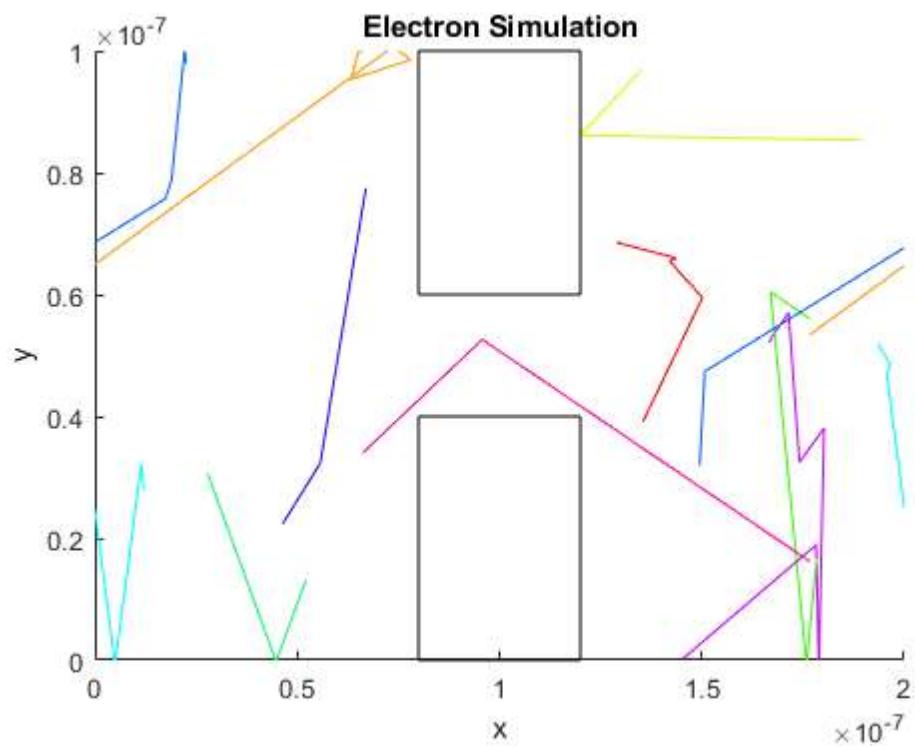


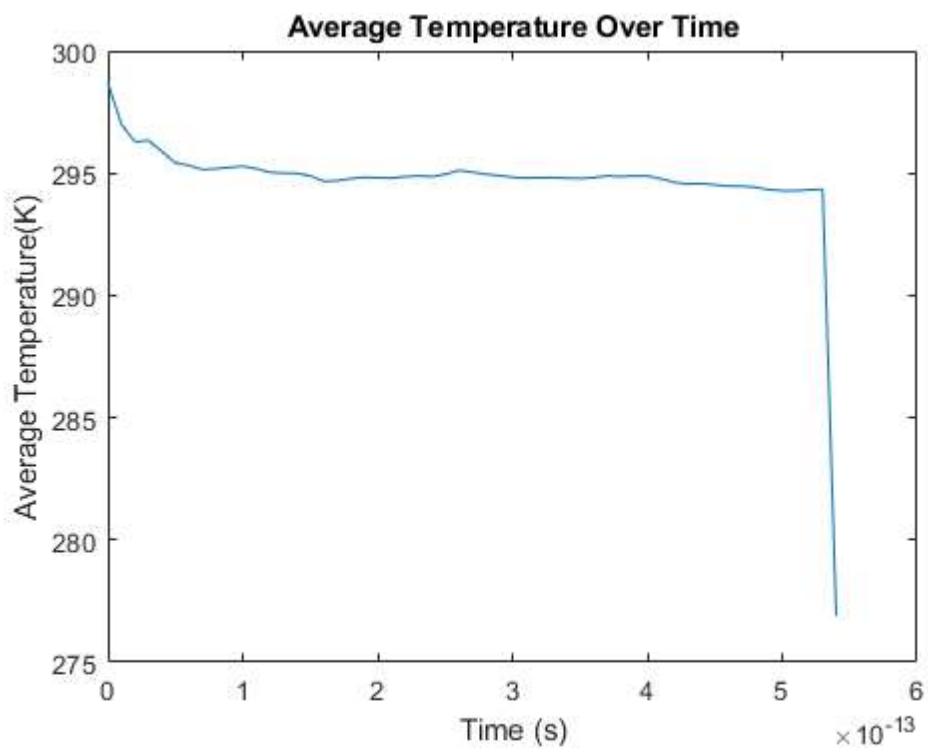
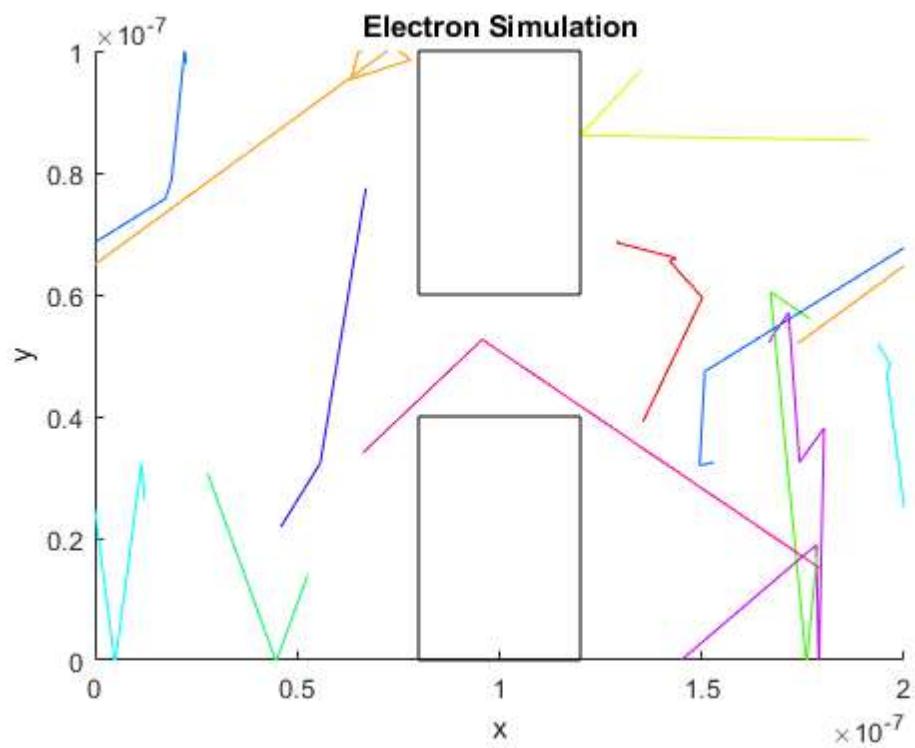


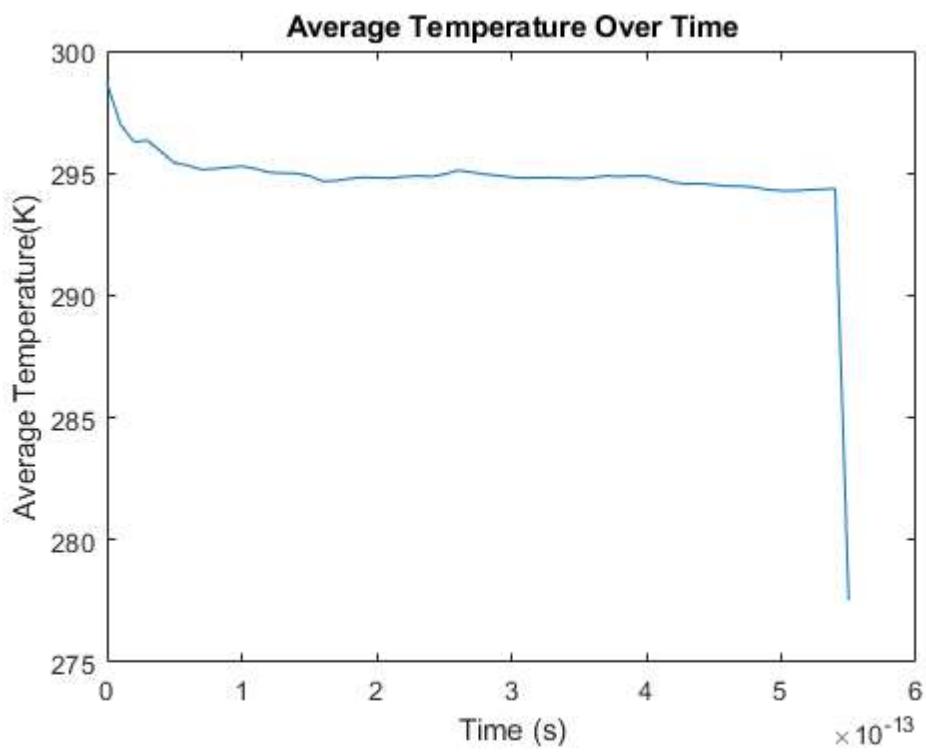
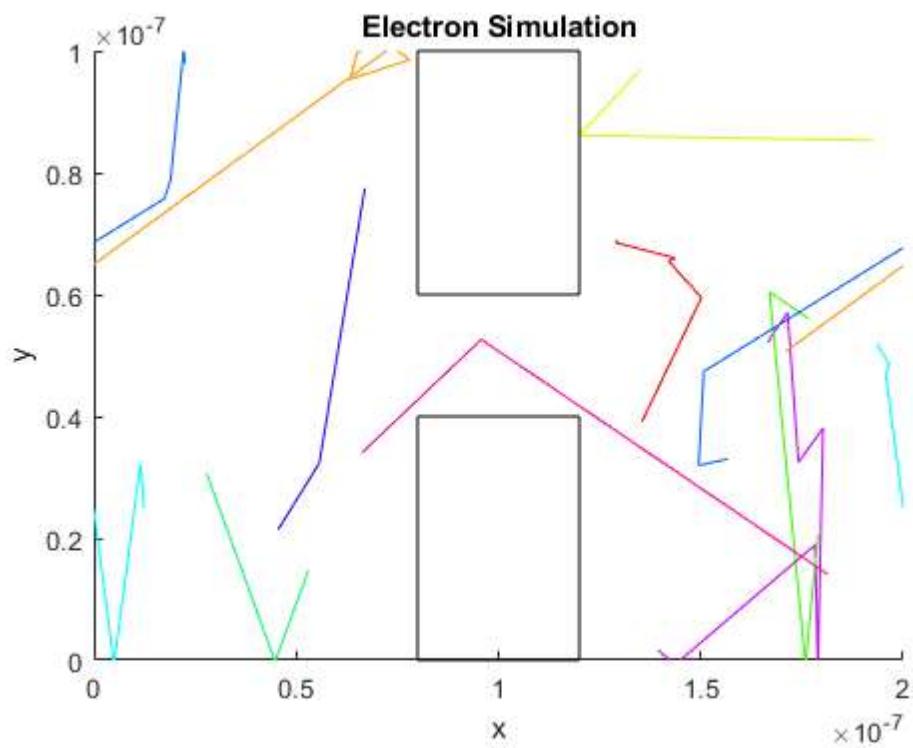


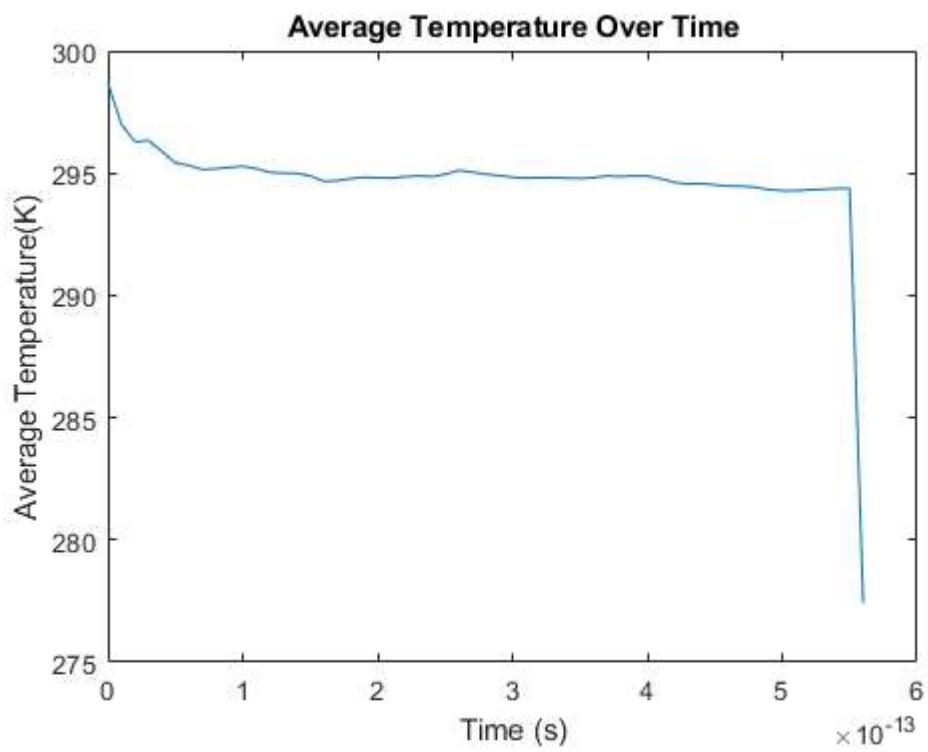
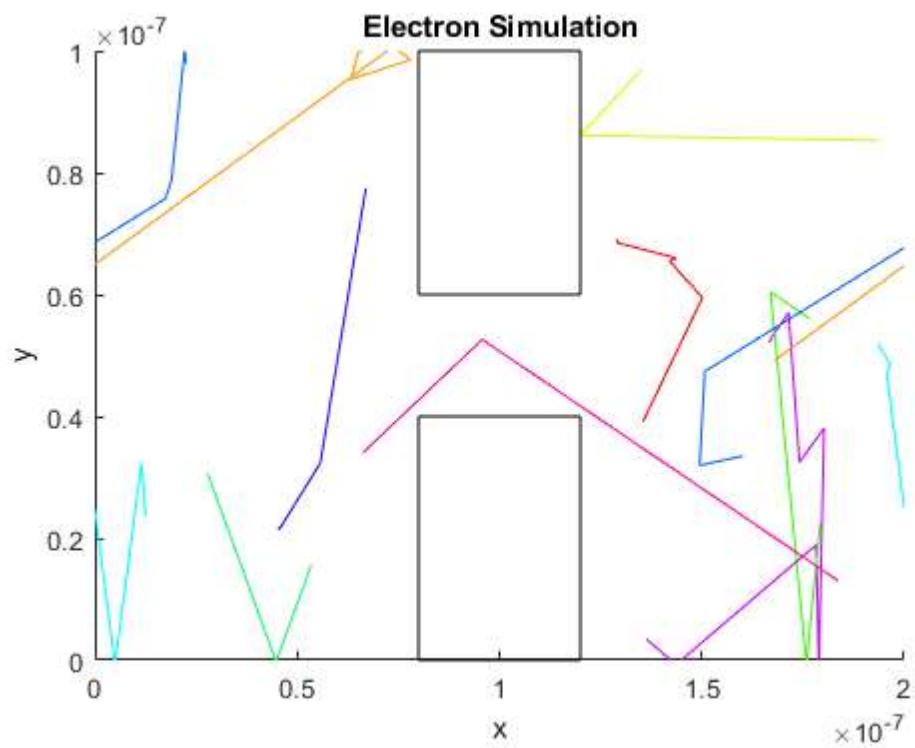


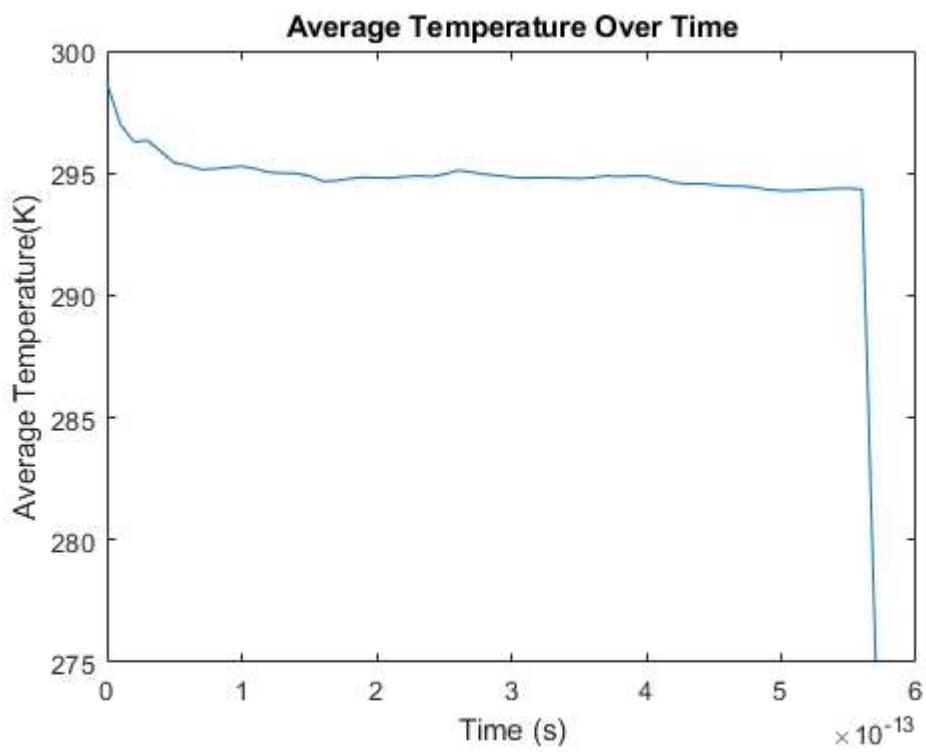
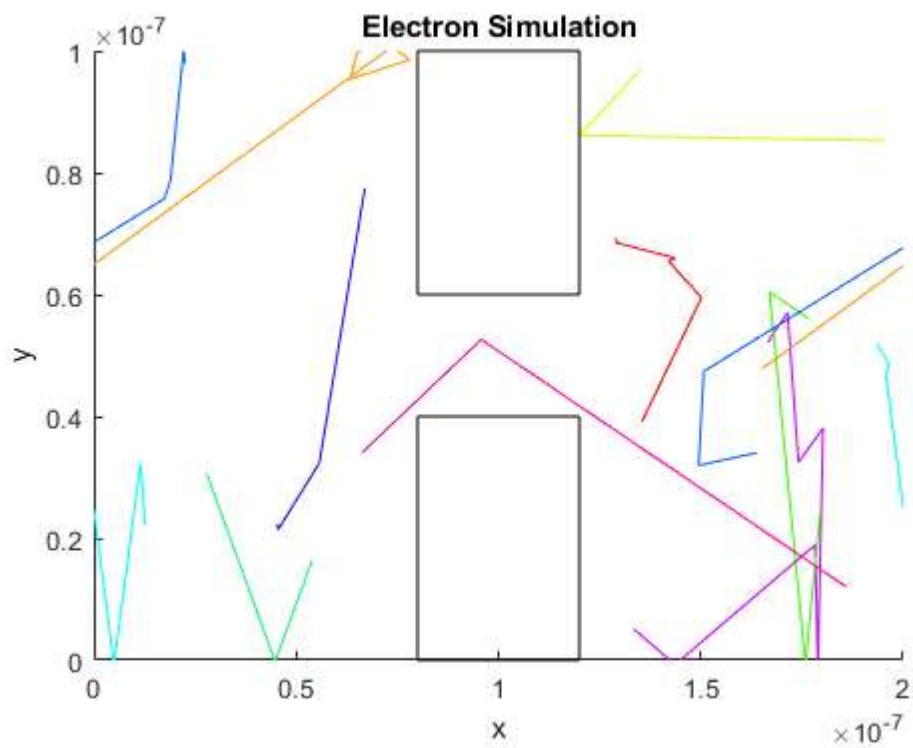


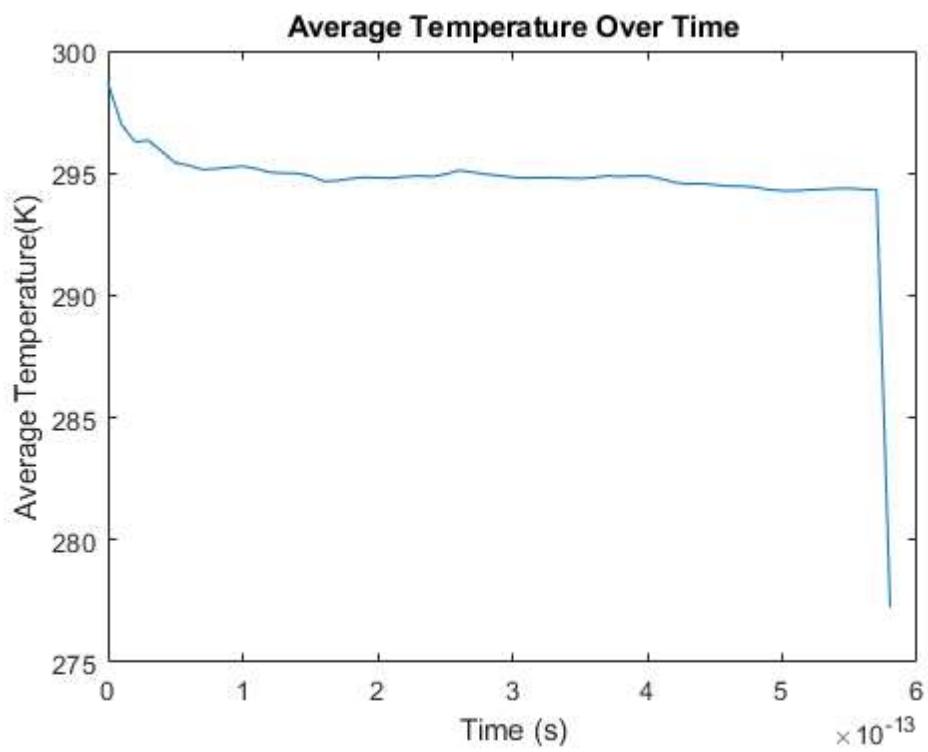
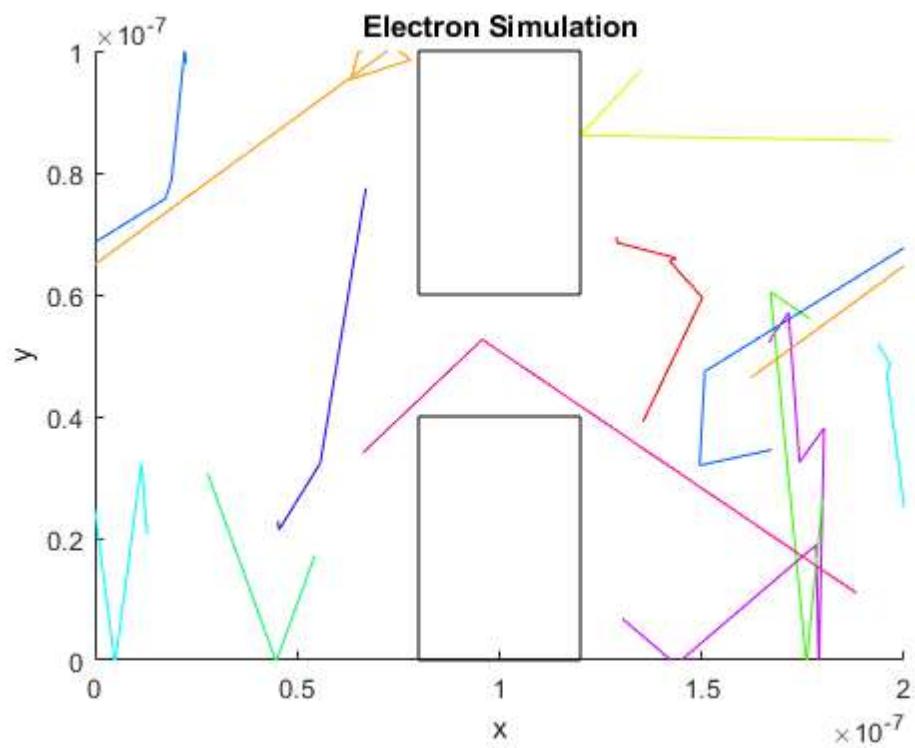


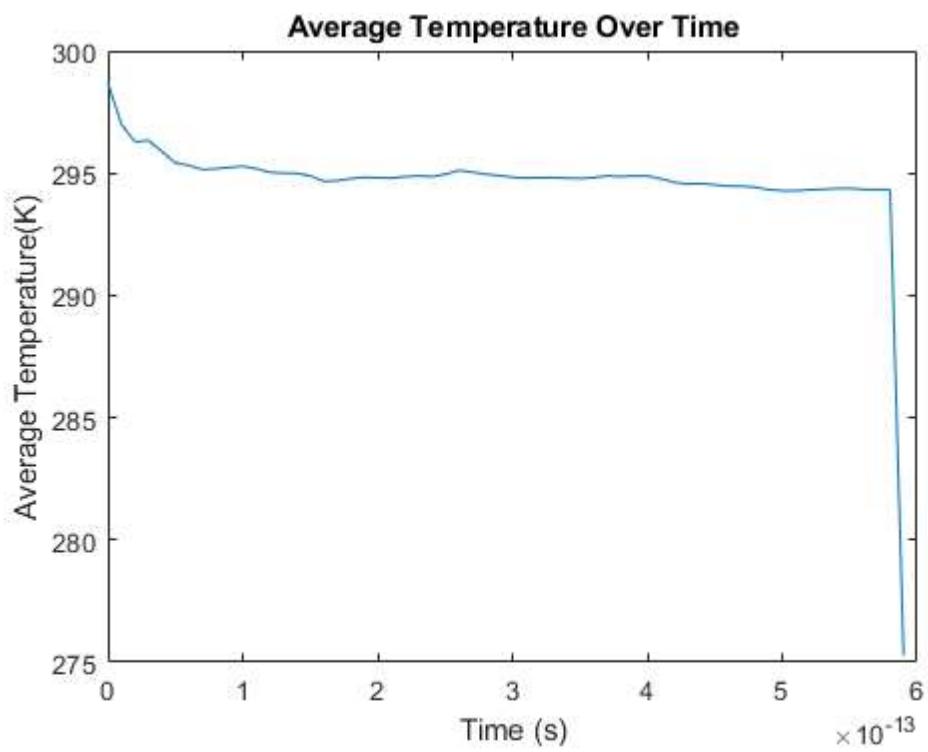
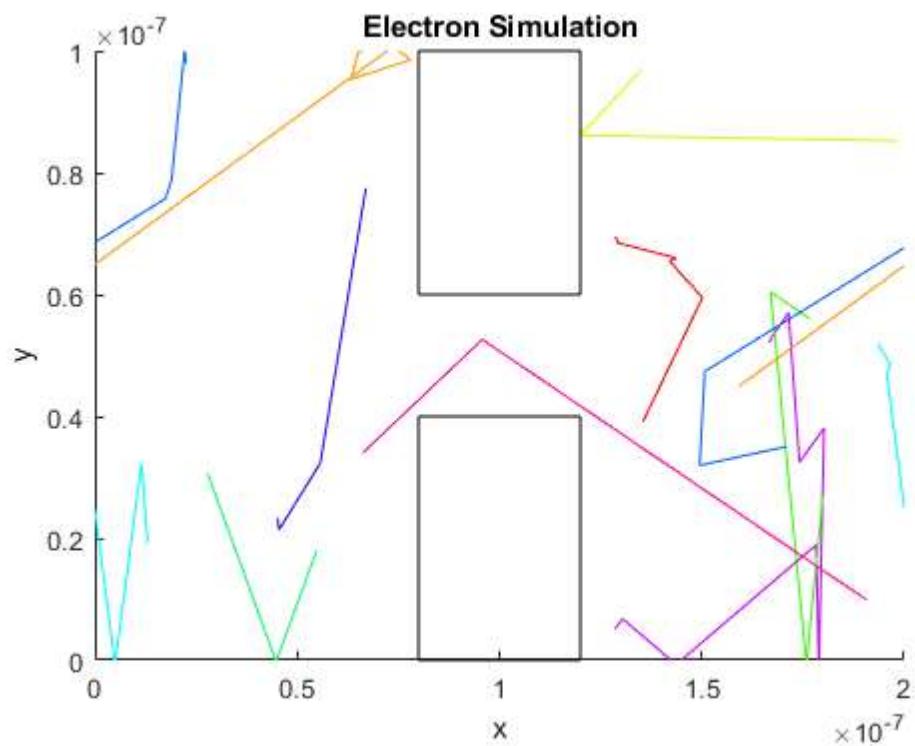


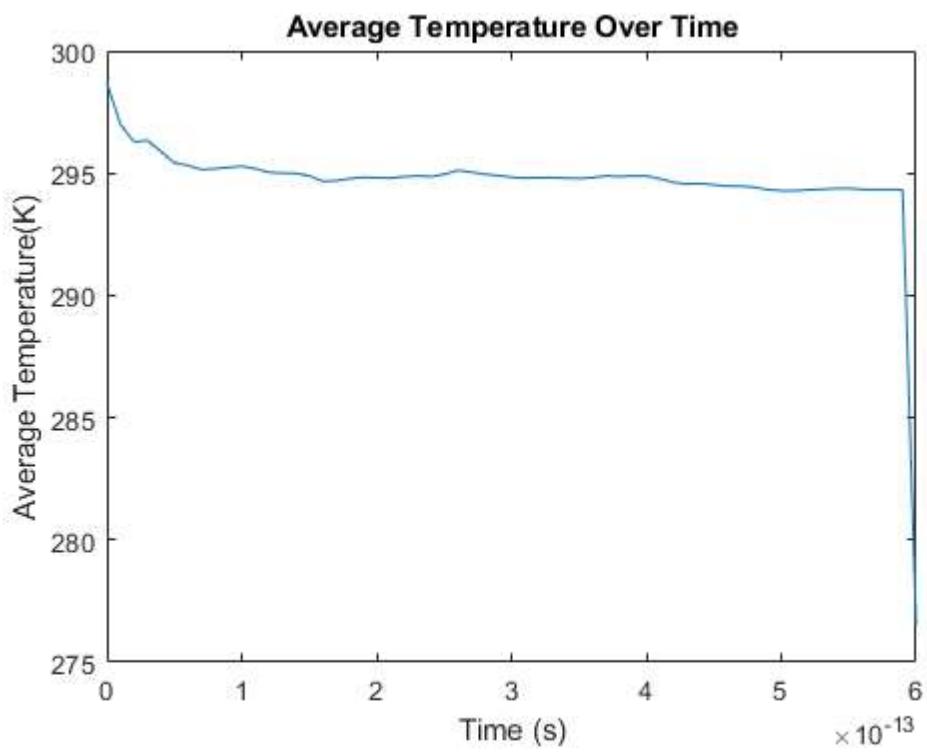
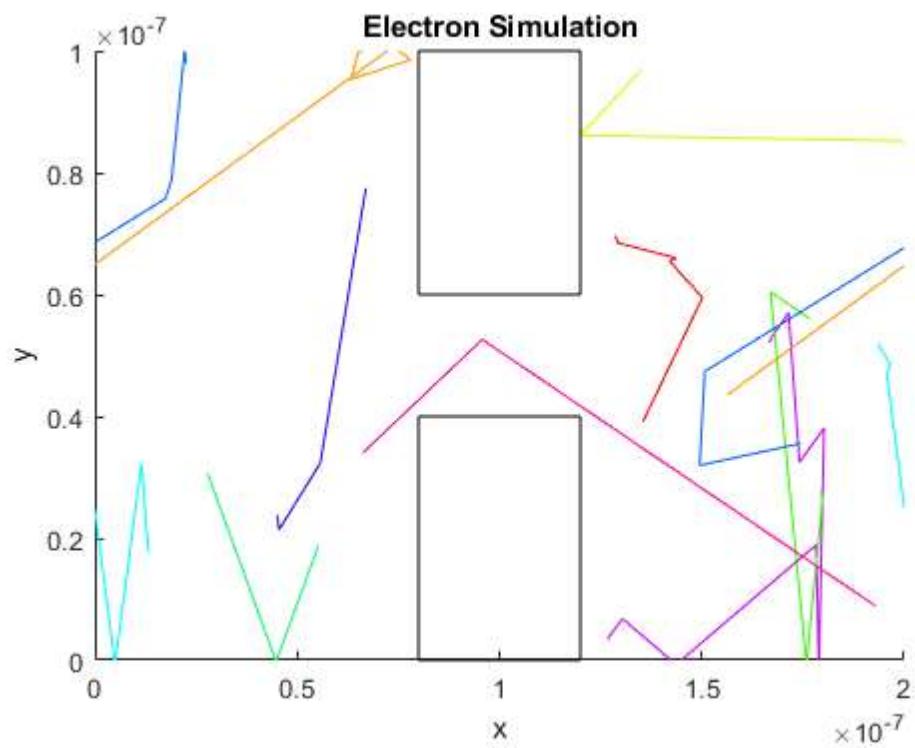


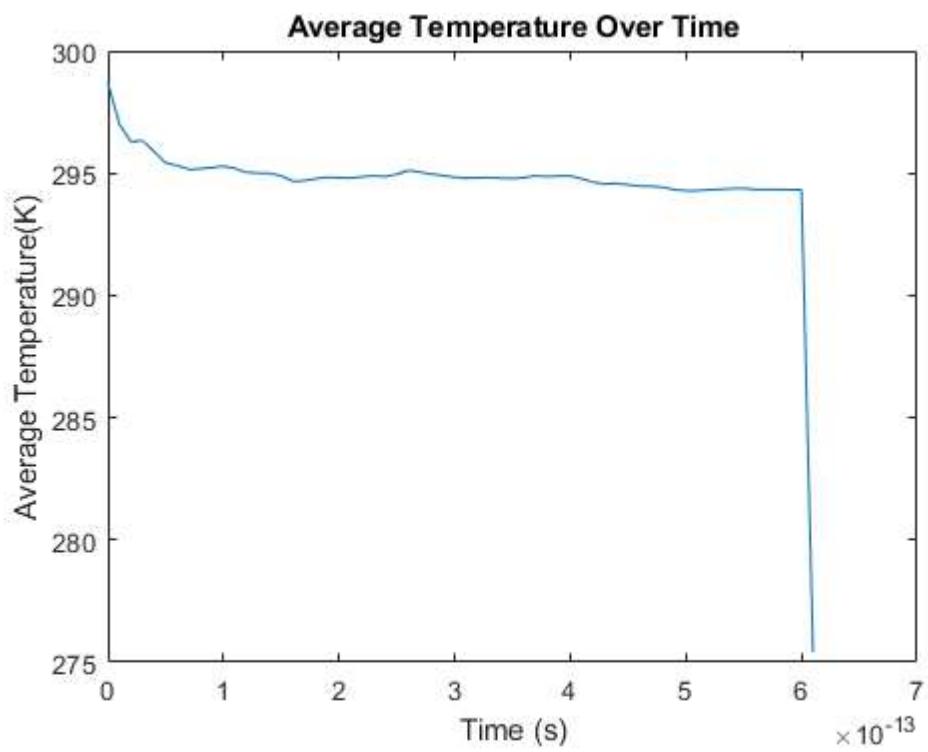
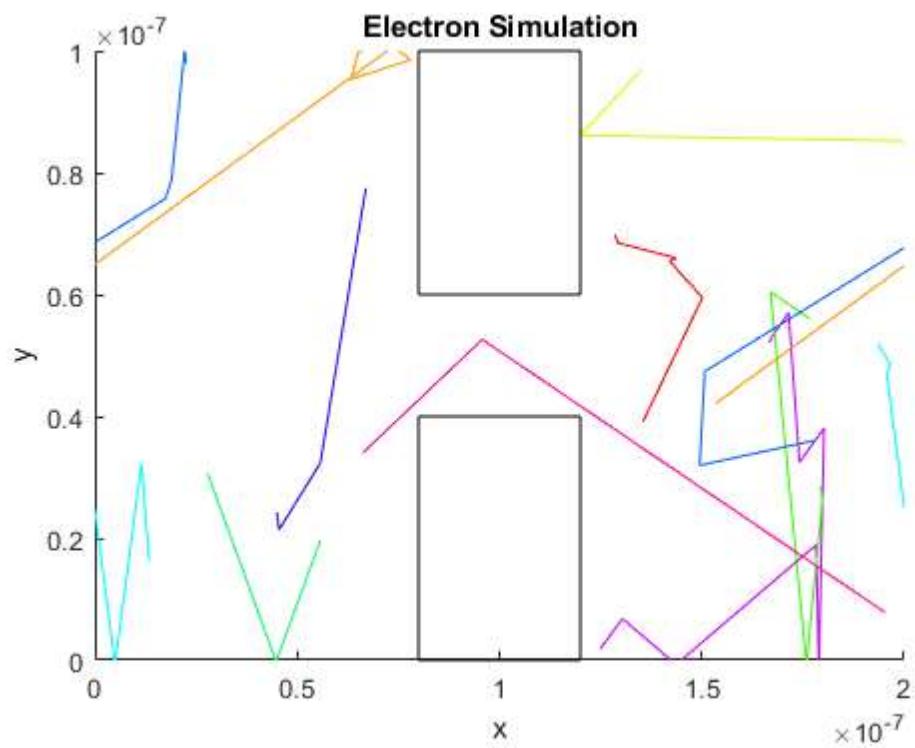


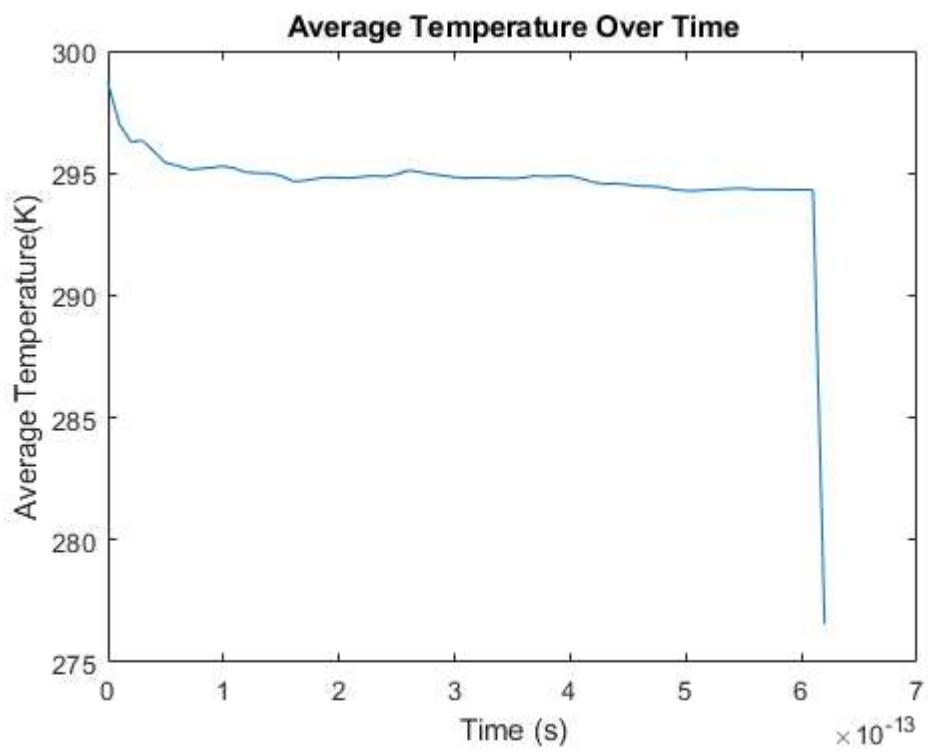
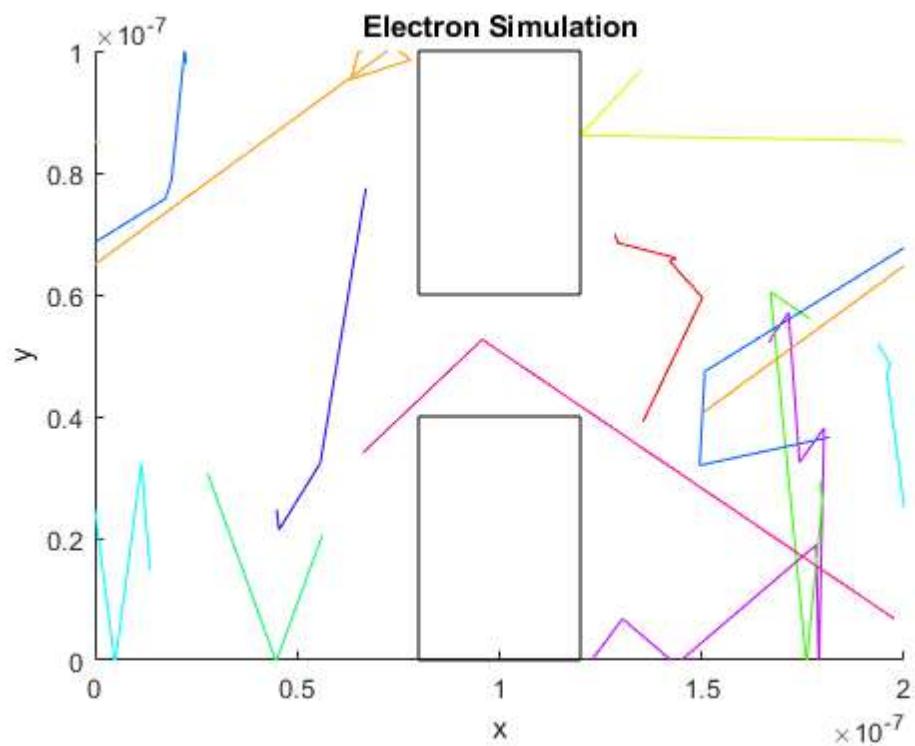


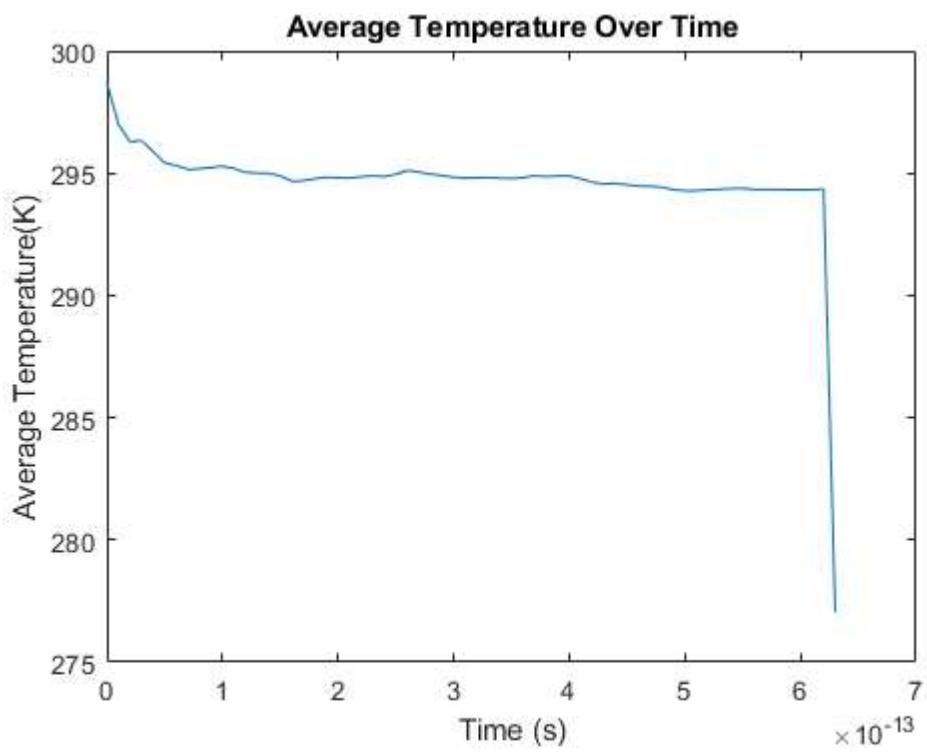
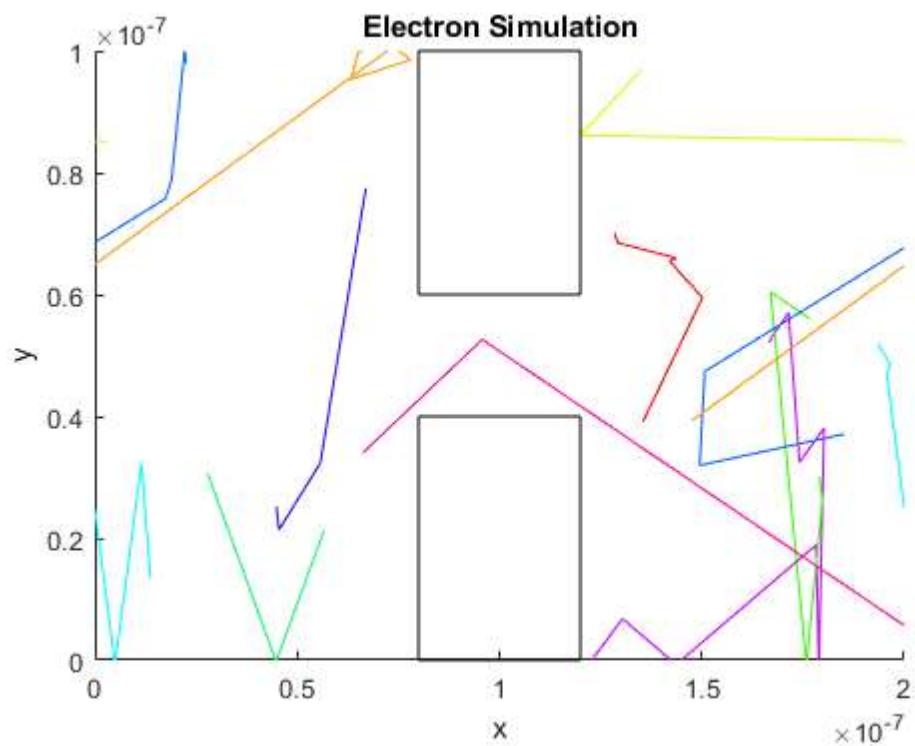


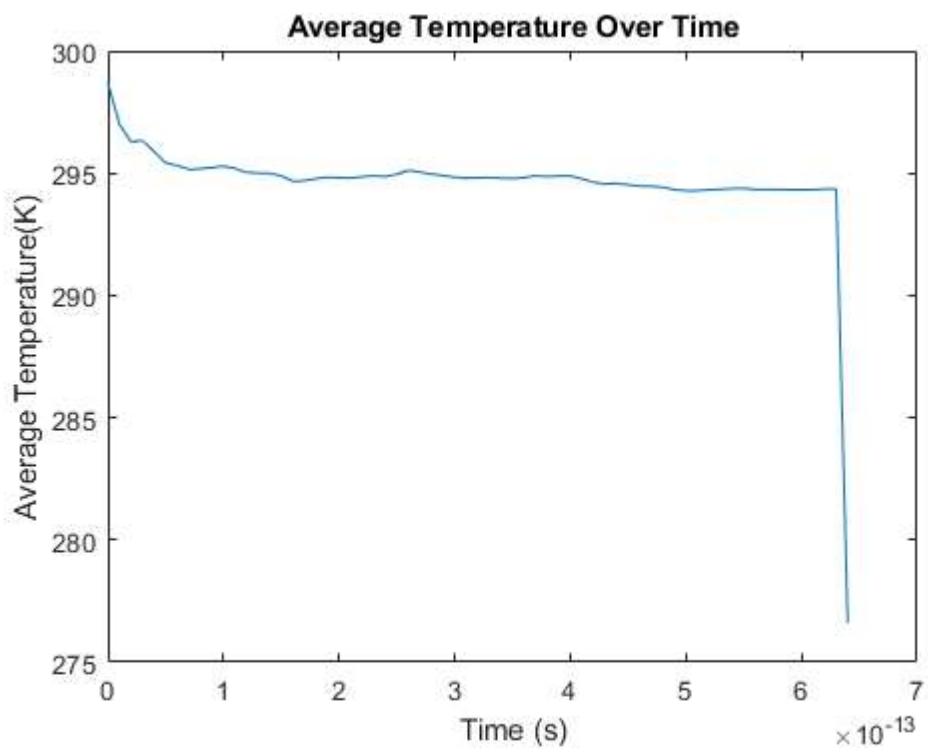
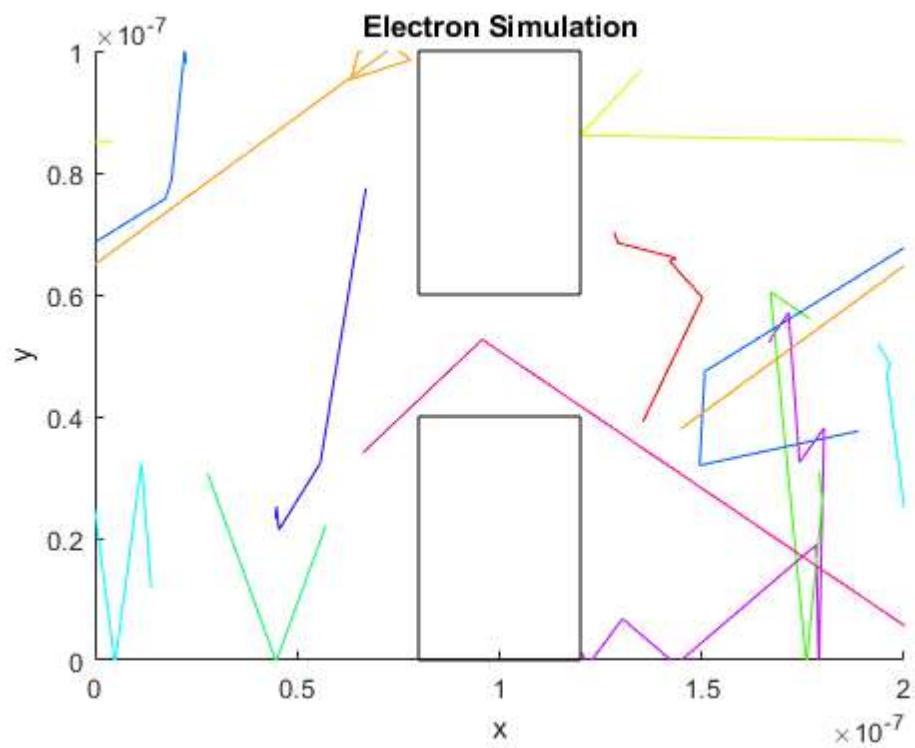


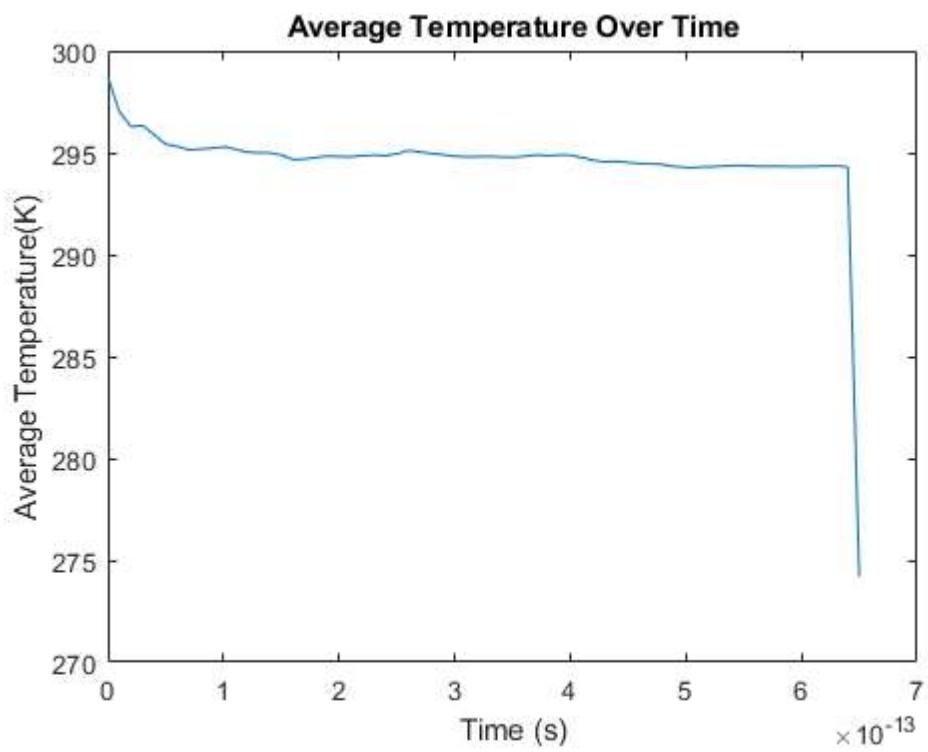
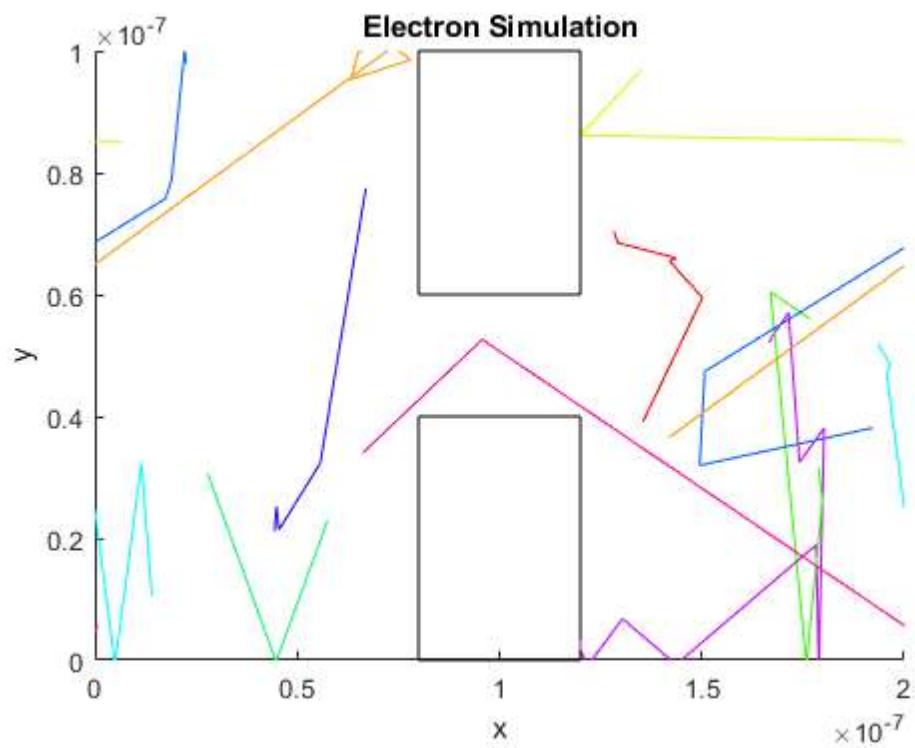


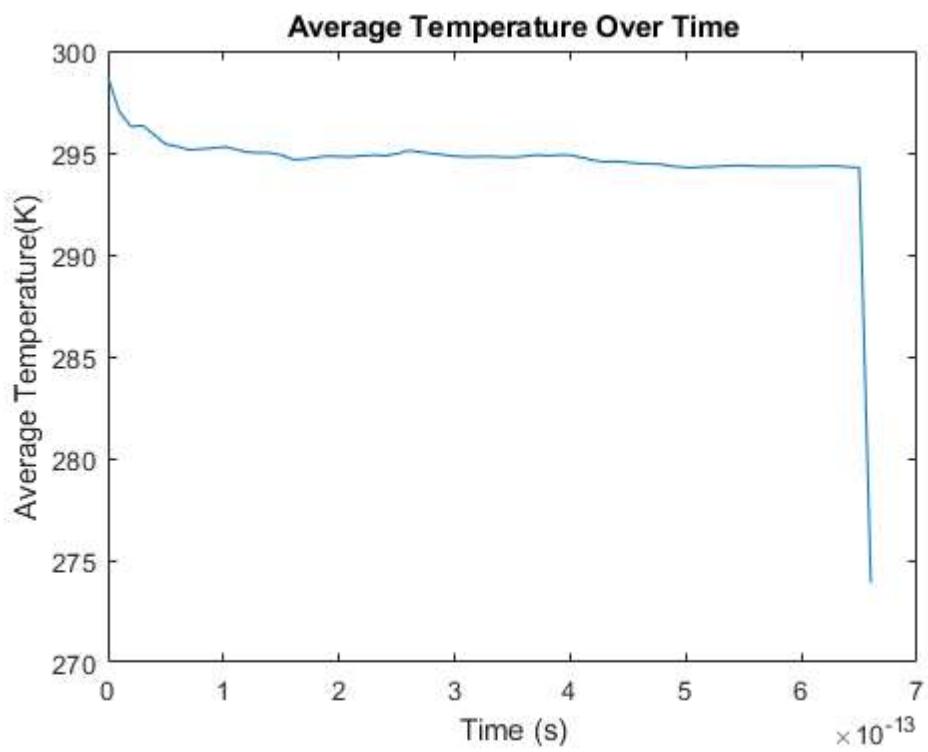
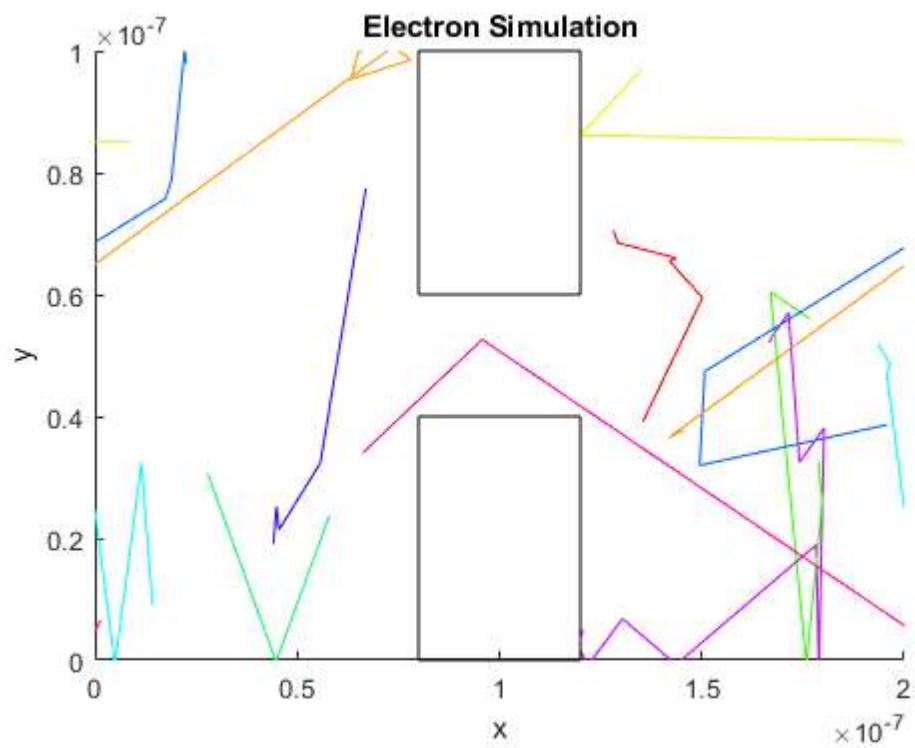


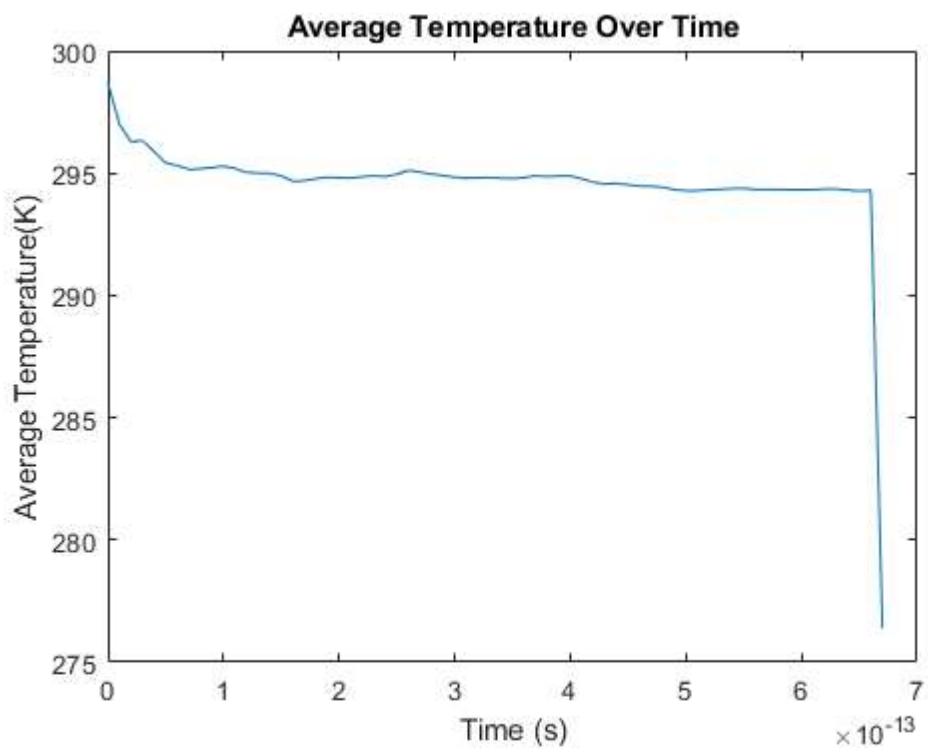
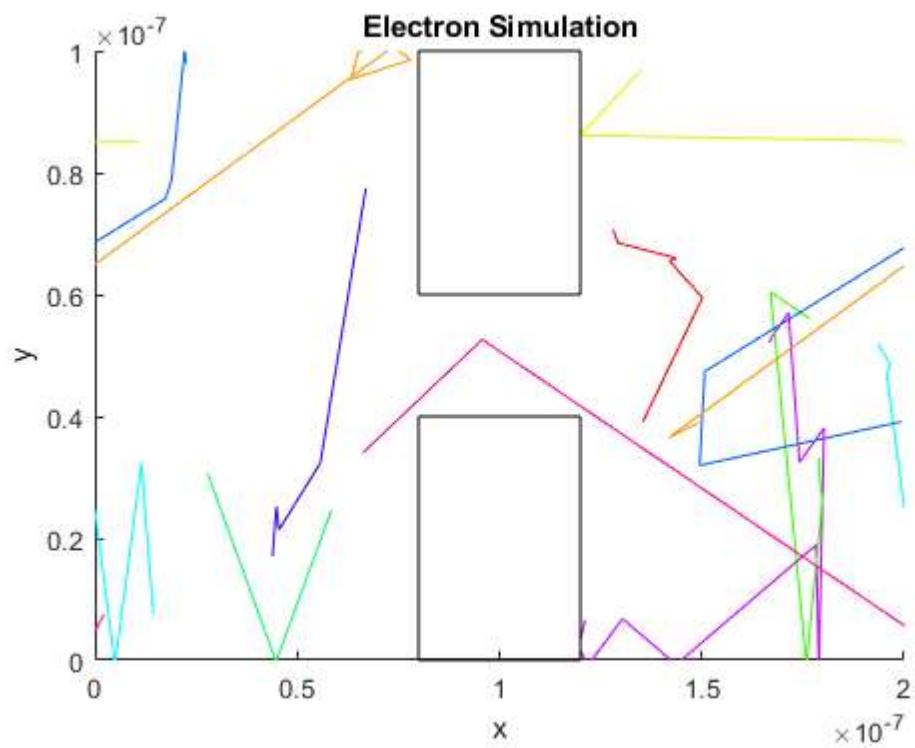


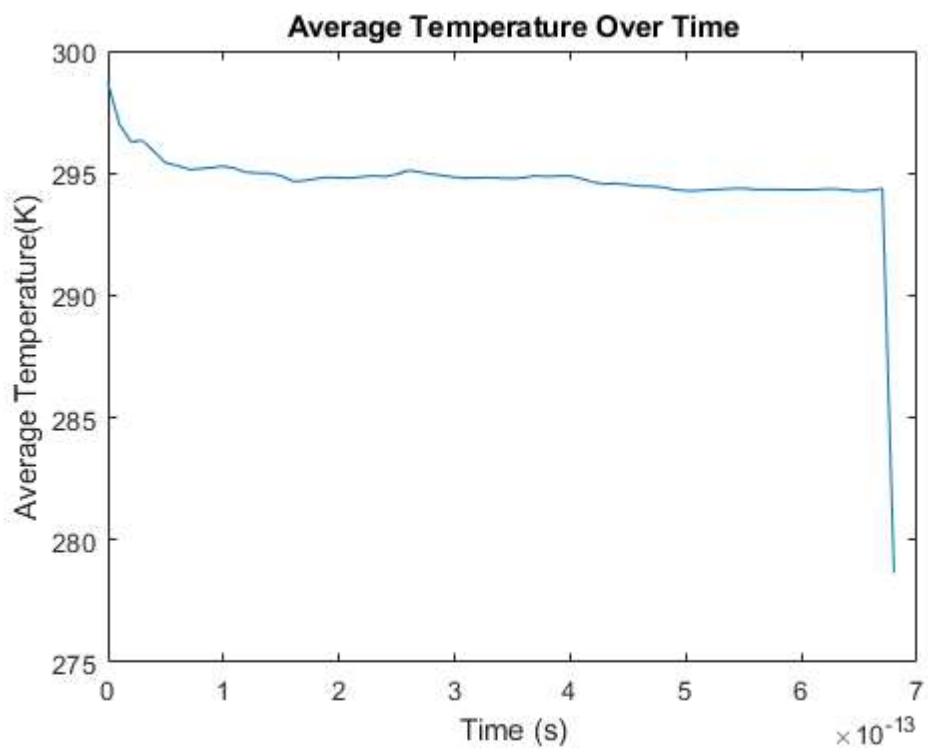
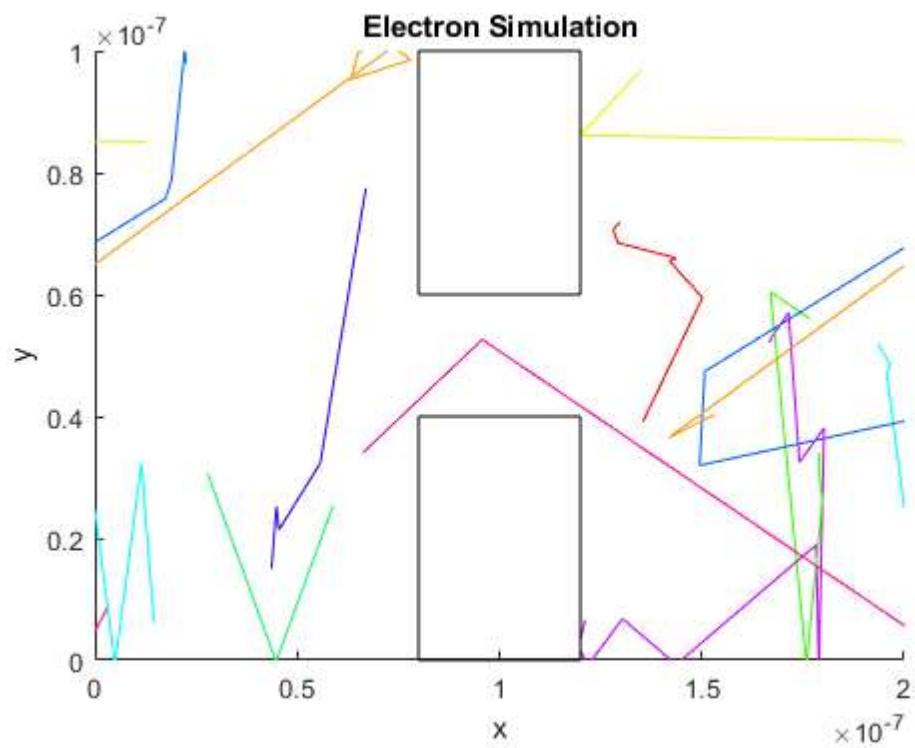


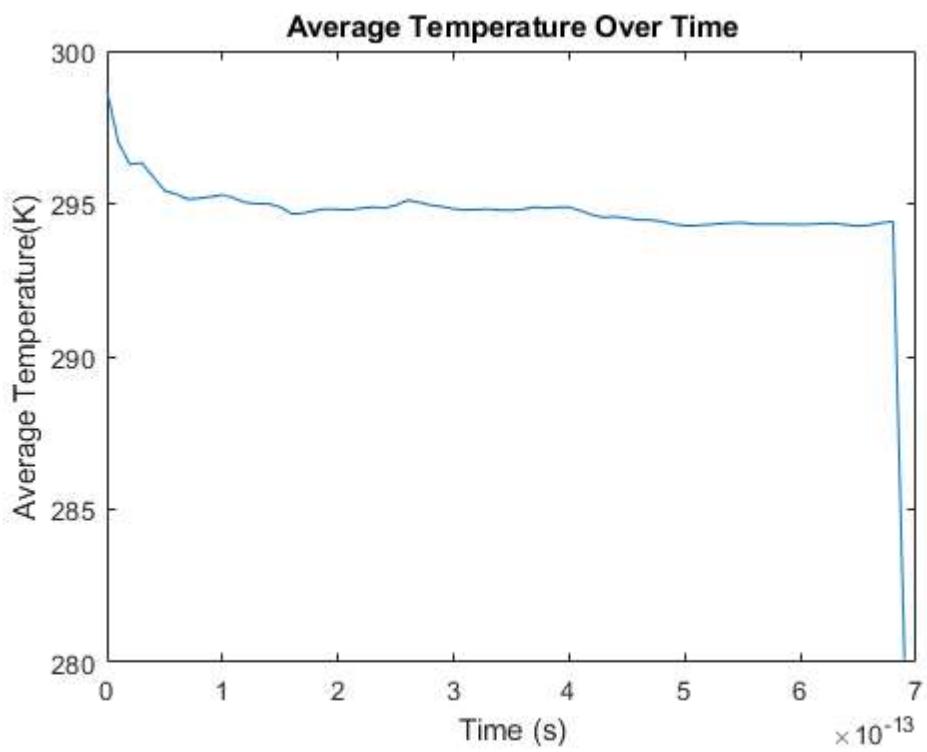
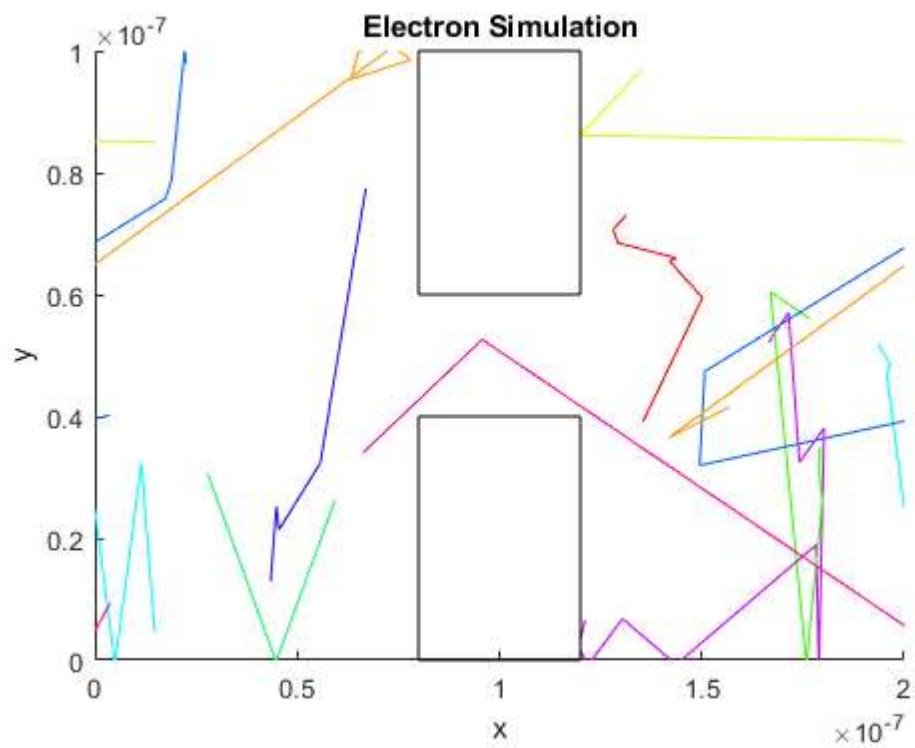


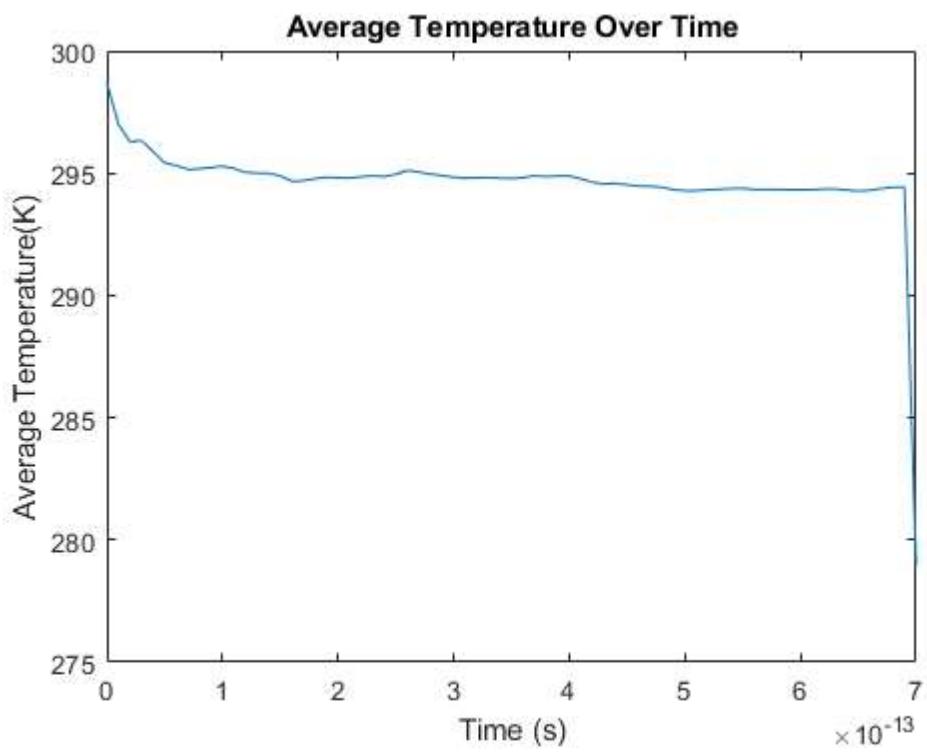
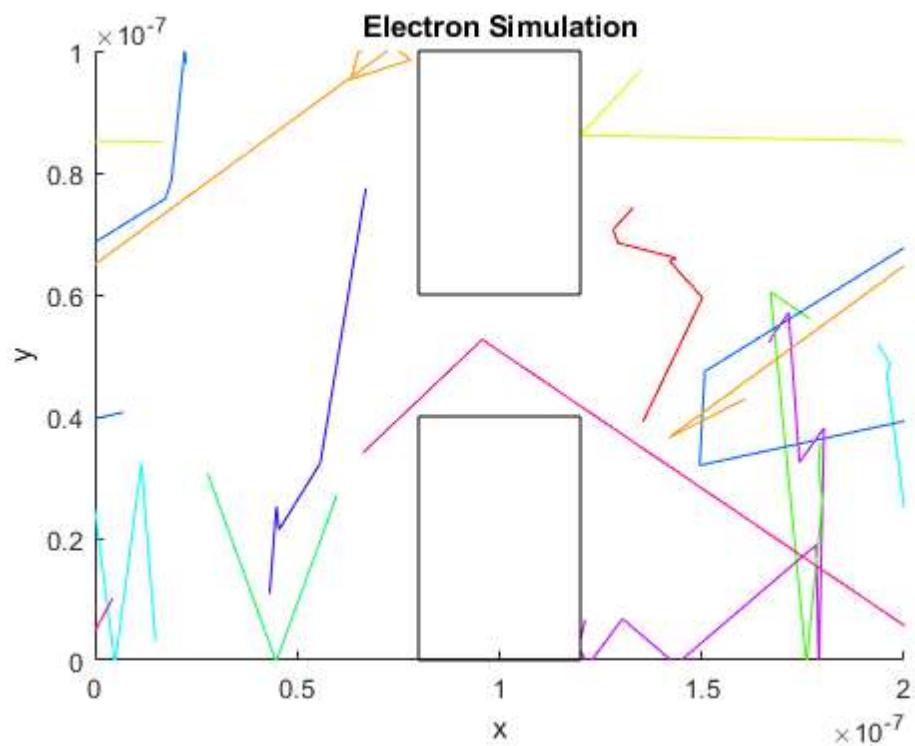


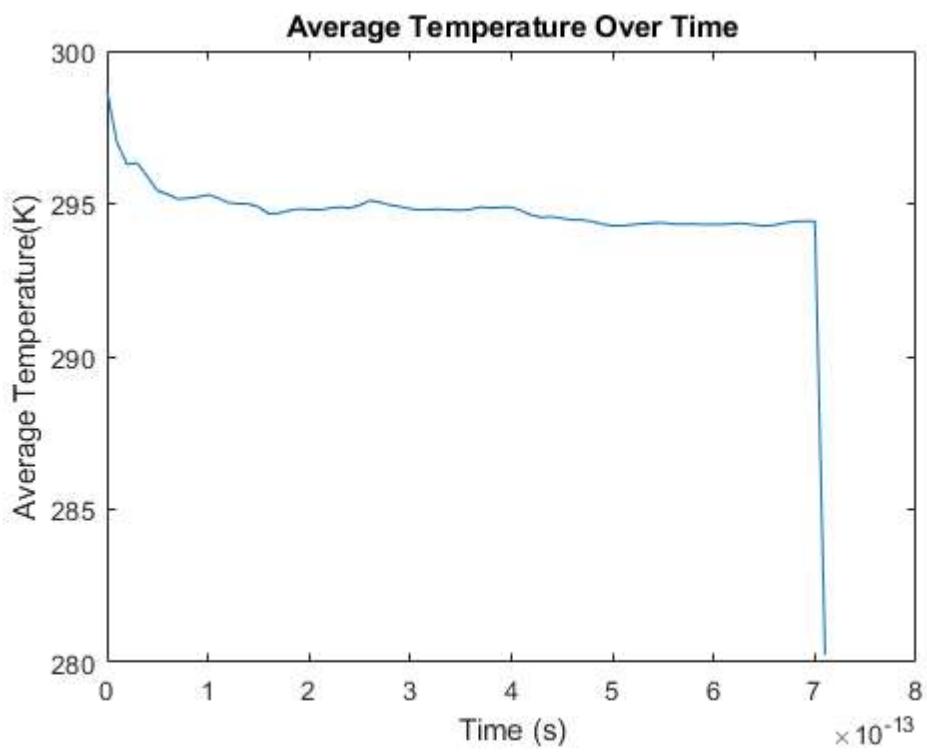
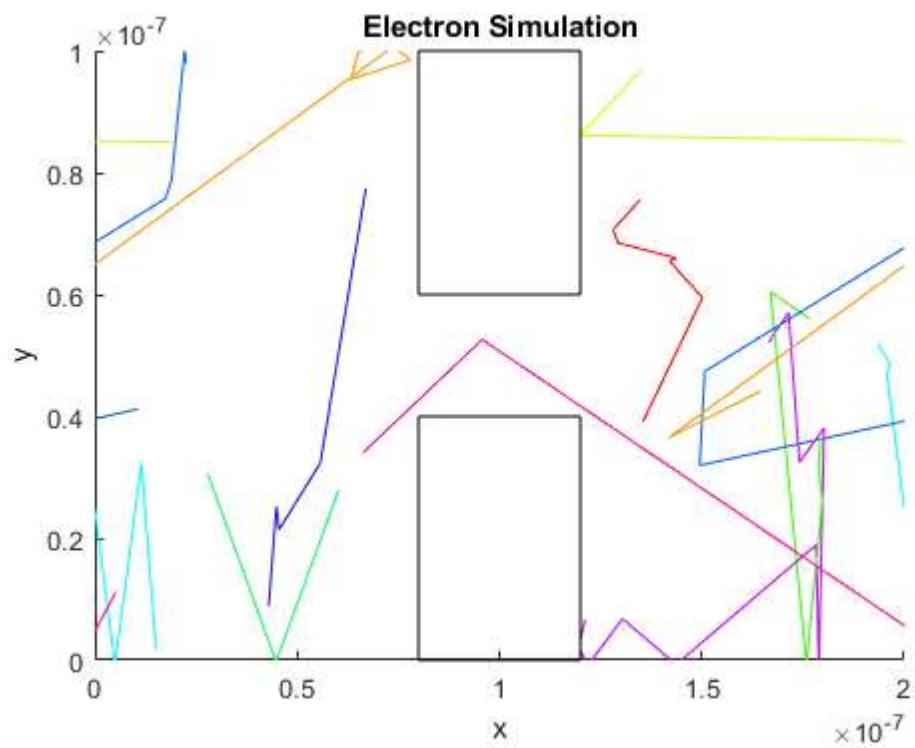


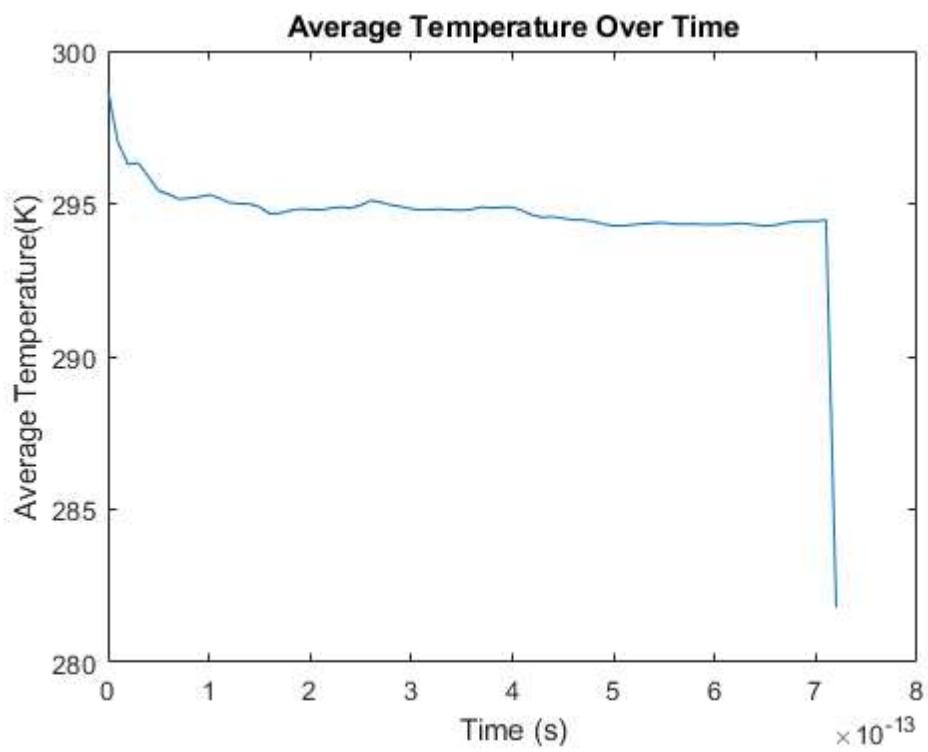
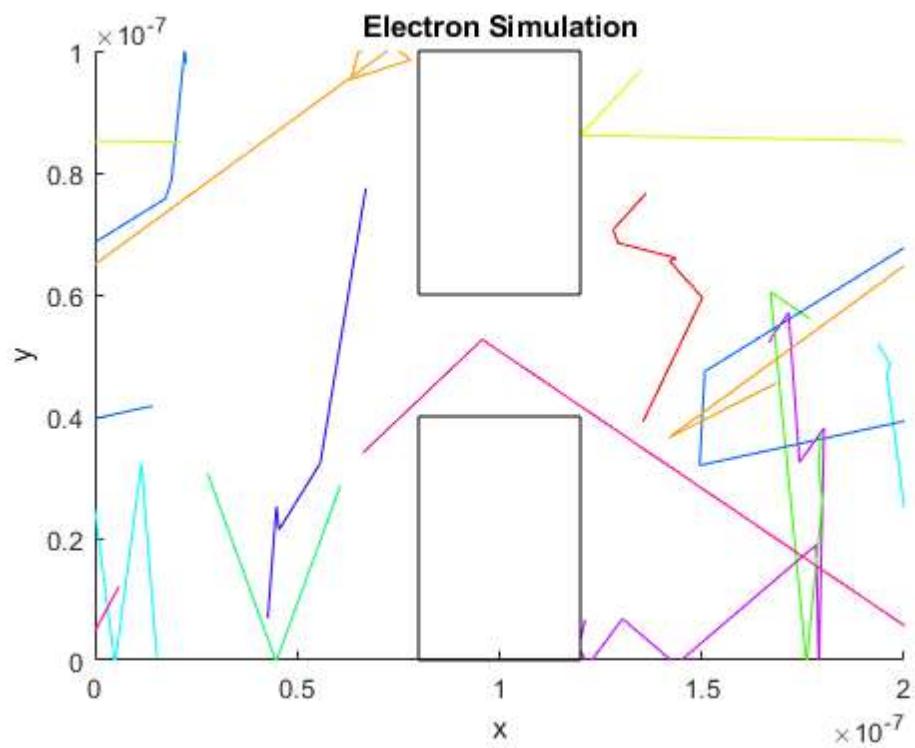


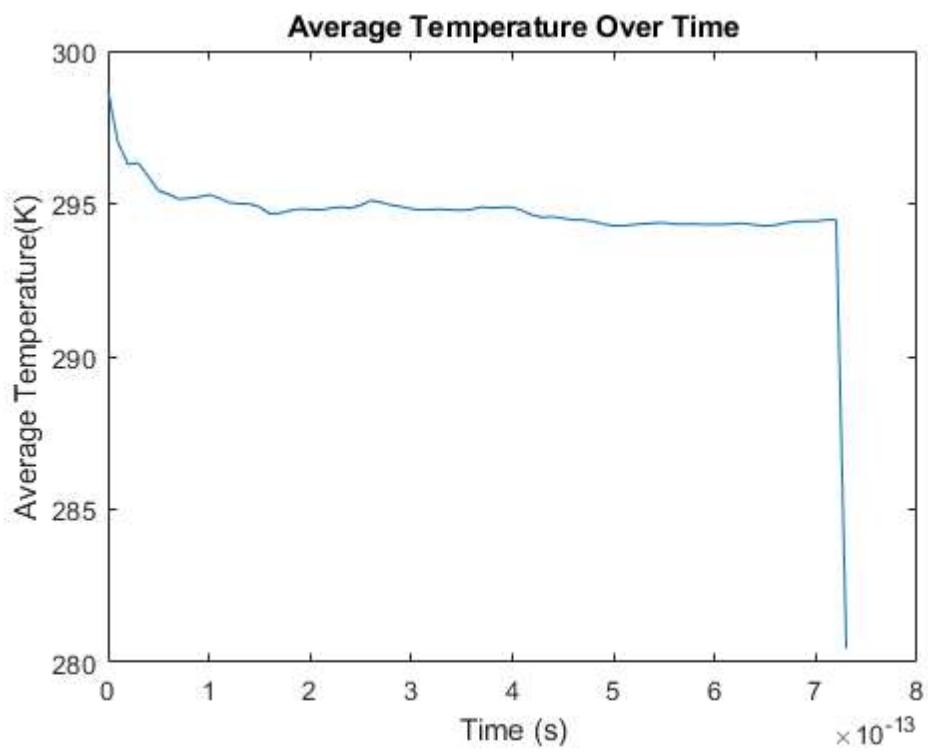
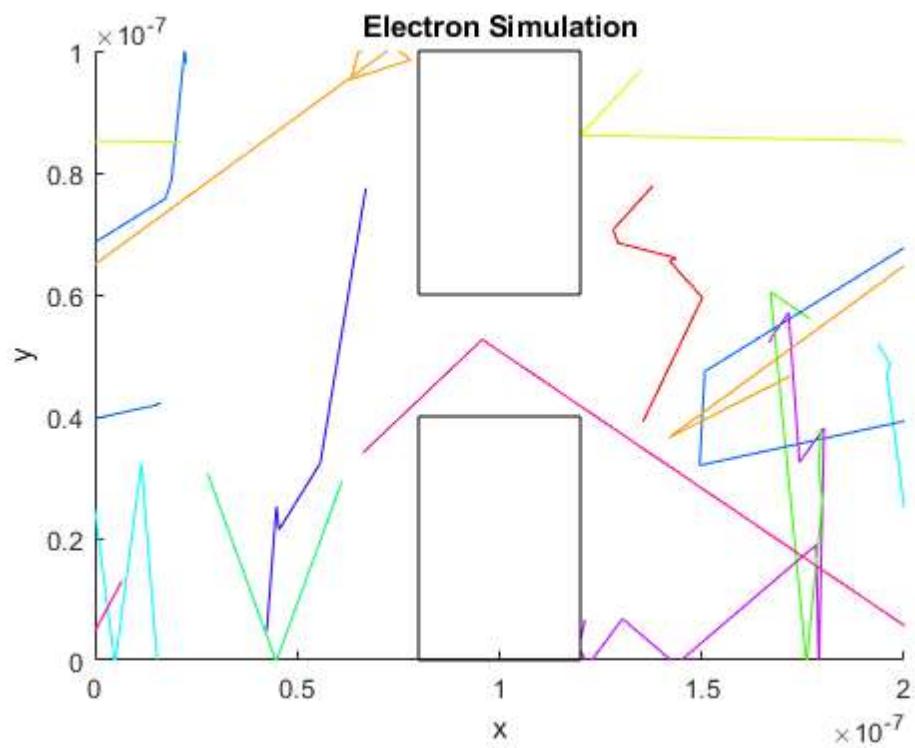


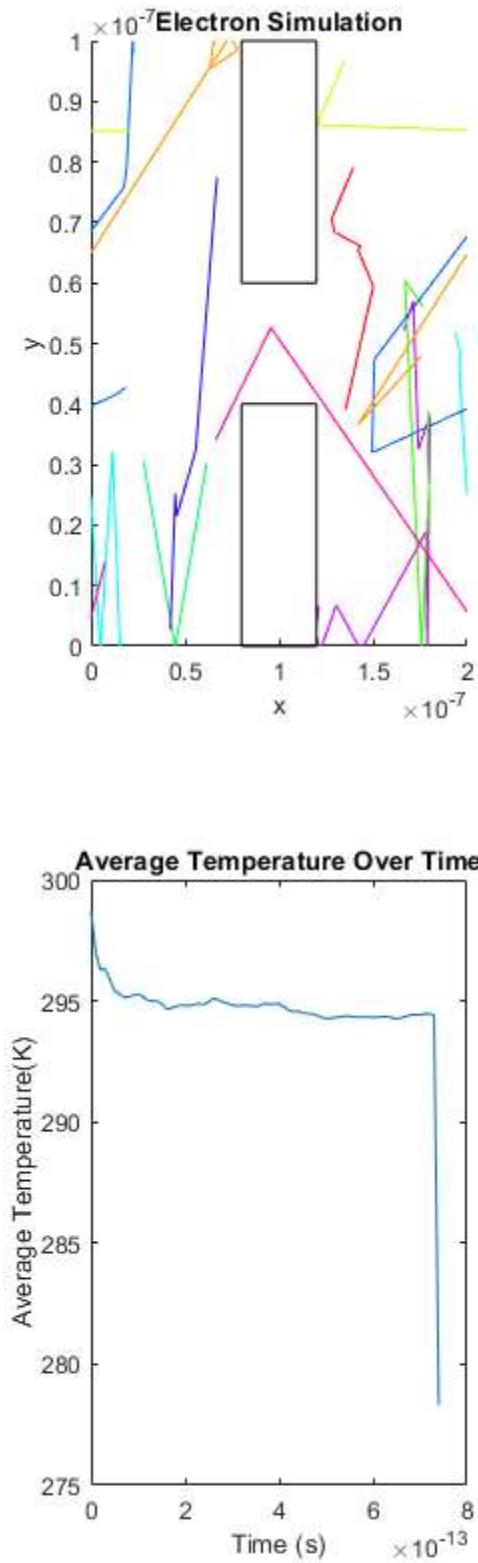
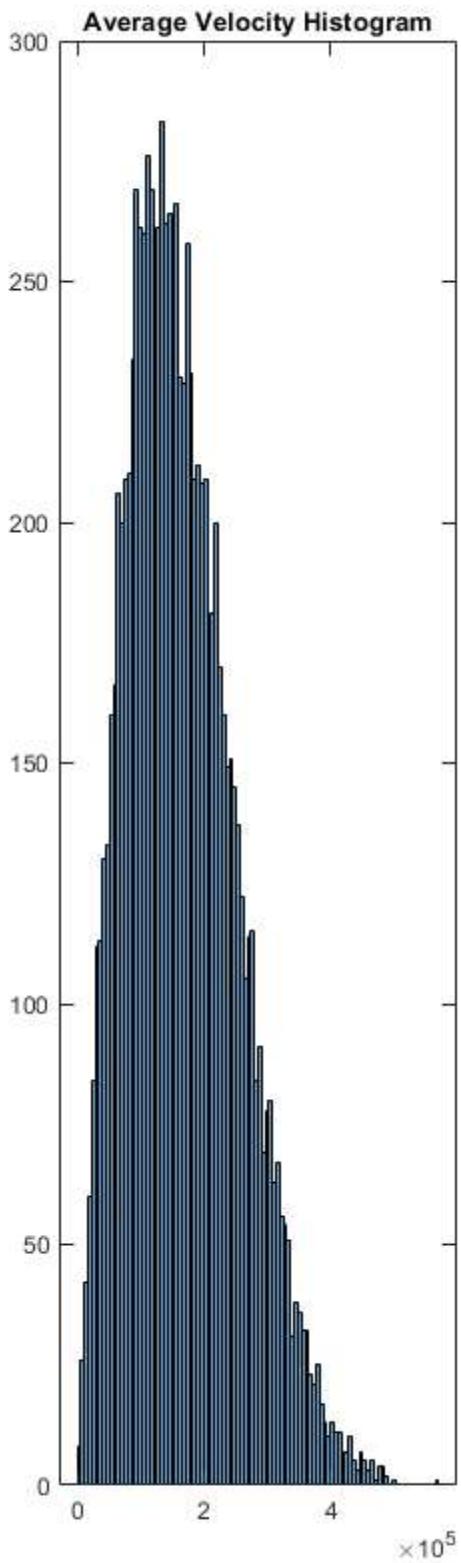












Then the average heat plot is updated.

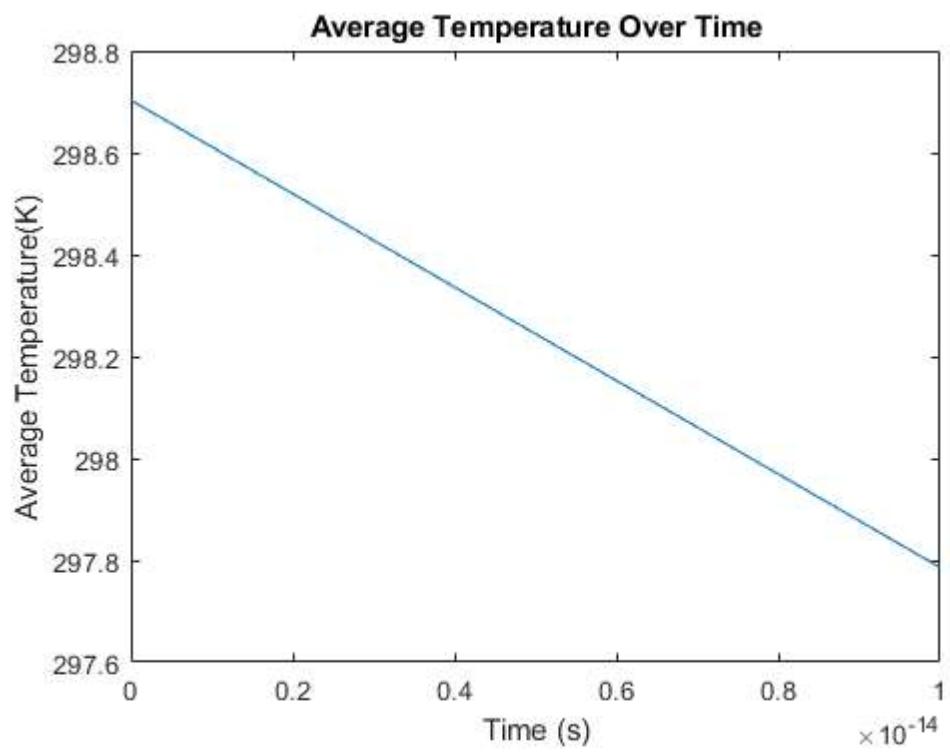
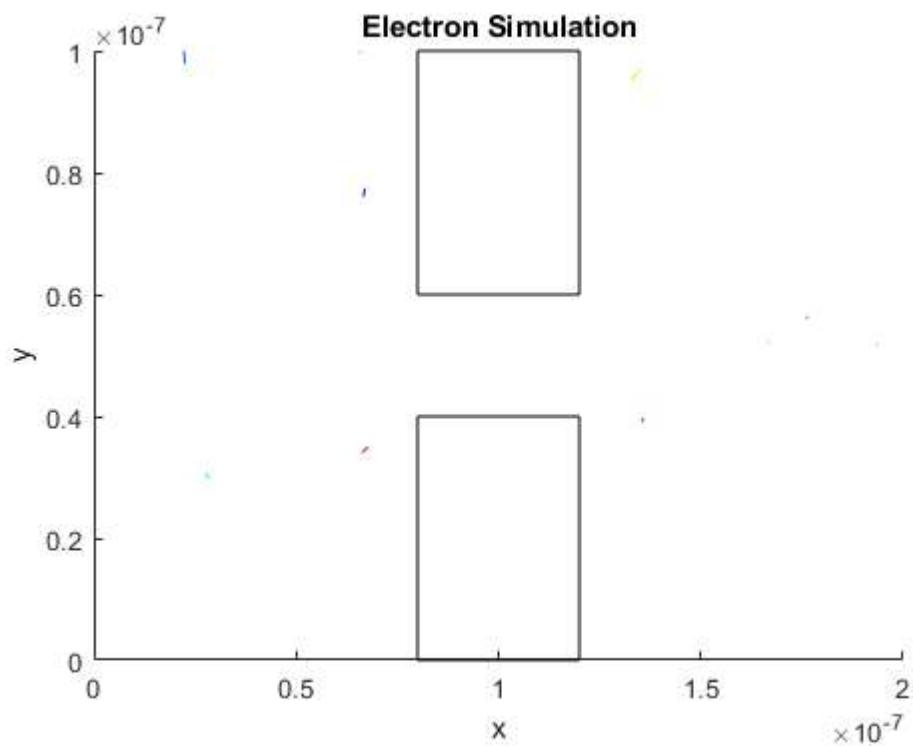
```
avev = sqrt(vy.*vy + vx.*vx);
avet = ((avev.*avev)*mn)/(2*kB);
asumavt = sum(avet)/NE;
sumavt = cat(1, sumavt, asumavt);
sumavt(i) = sum(sumavt)/length(sumavt);
subplot(2,1,2)
plot(time, sumavt);
xlabel('Time (s)')
ylabel('Average Temperature(K)')
title('Average Temperature Over Time');

ylt = y < 0;
ygt = y > h;

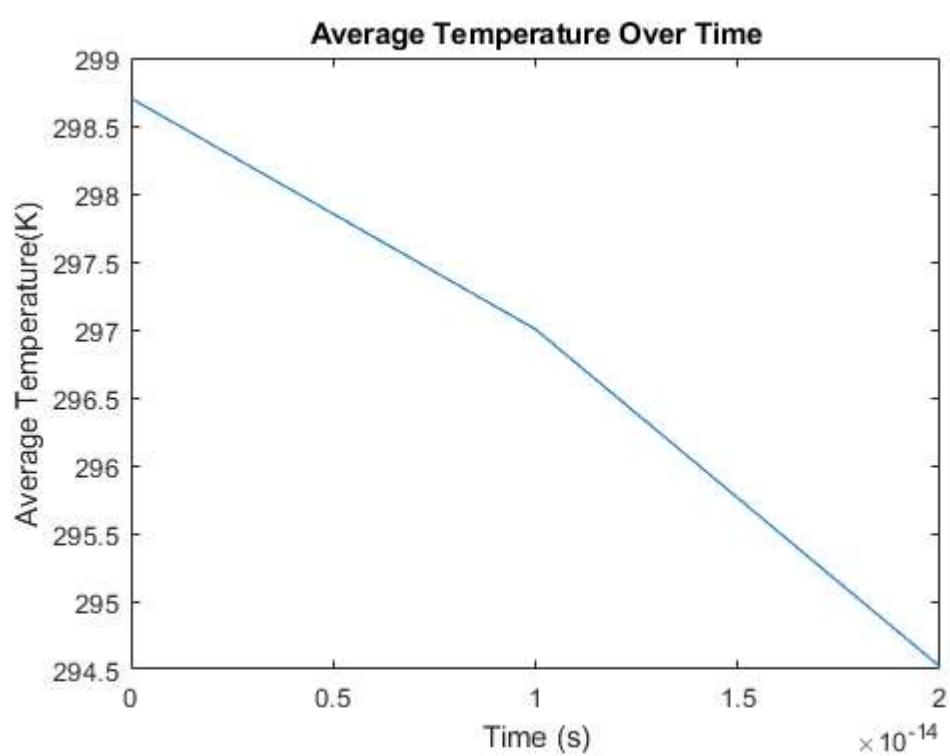
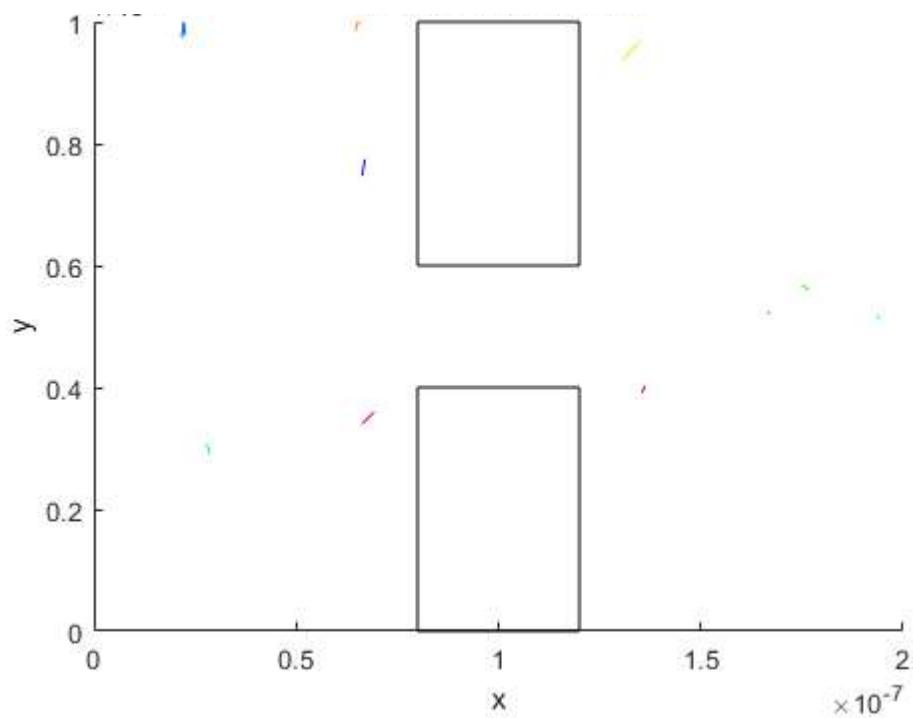
xlt = x < 0;
xgt = x > w;

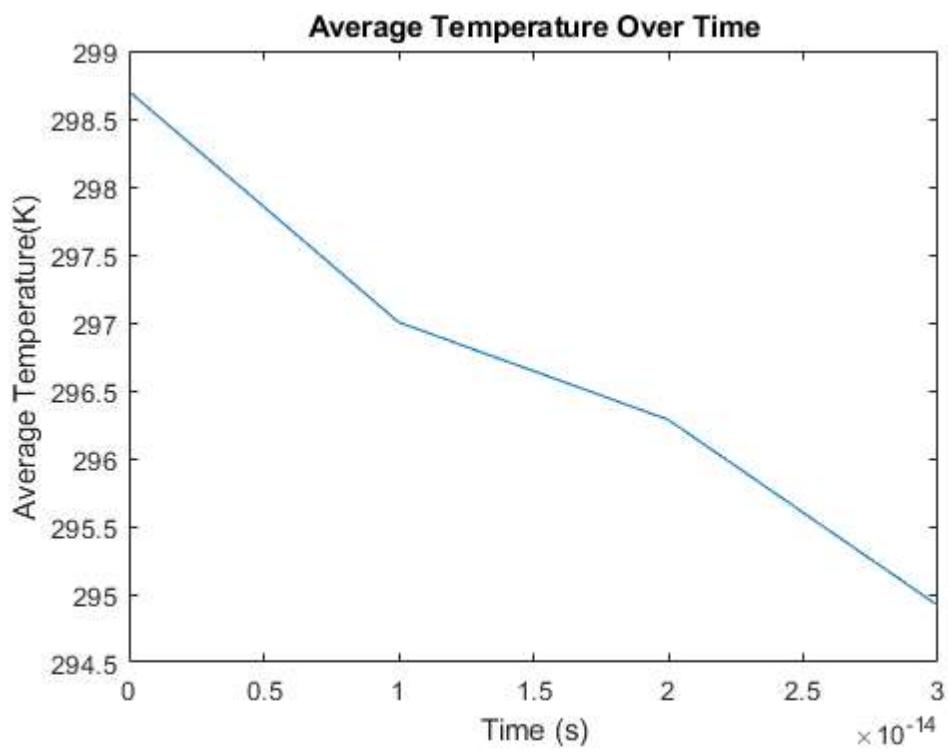
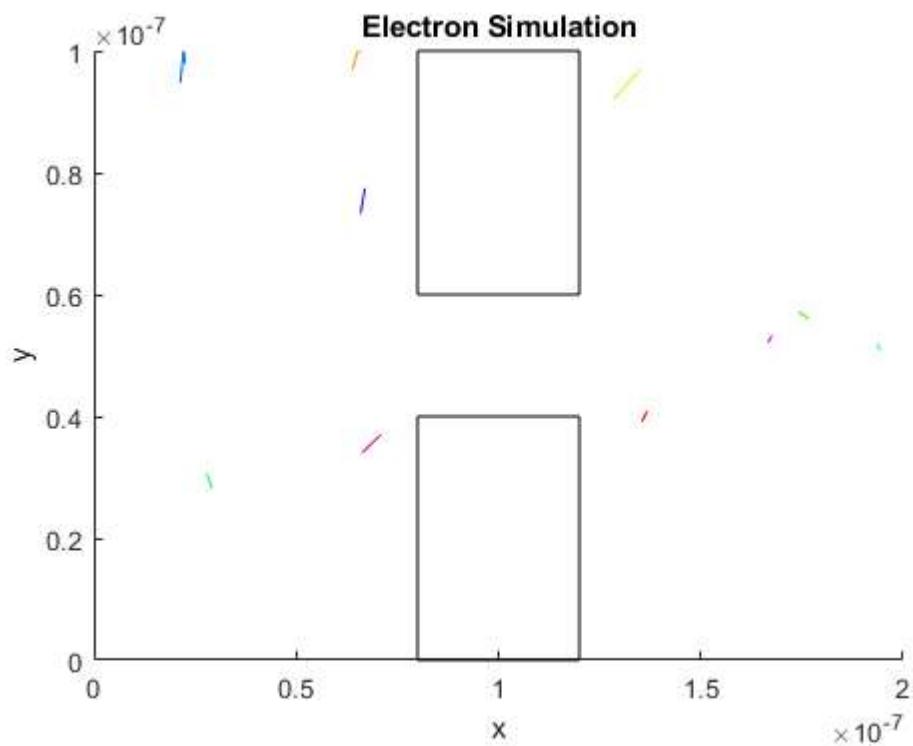
y(ylt) = 0;
y(ygt) = h;
x(xlt) = w;
x(xgt) = 0;

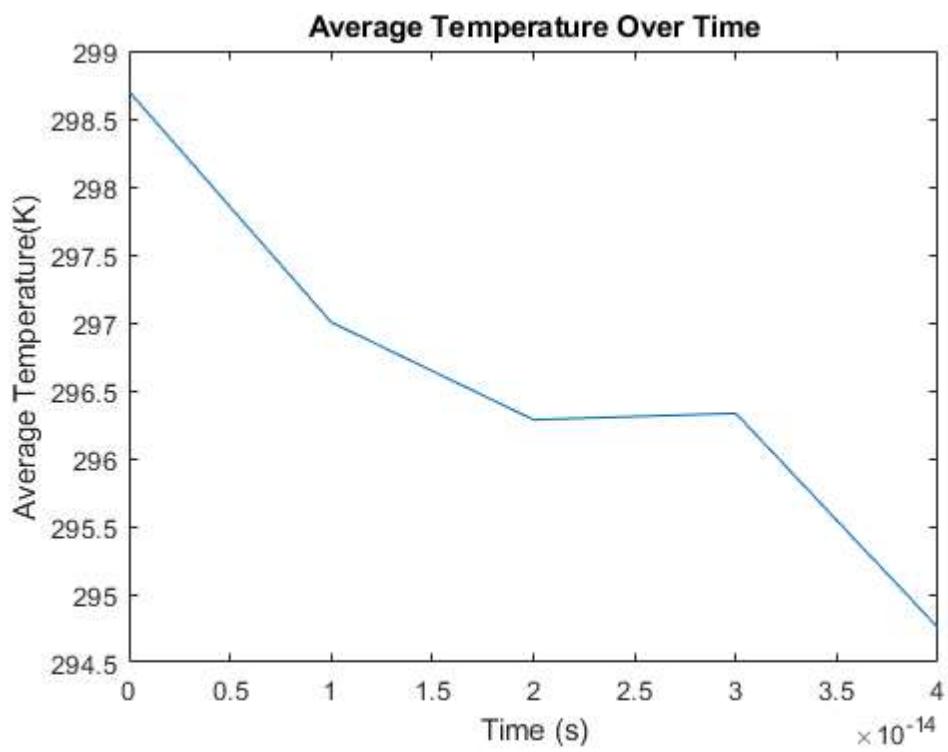
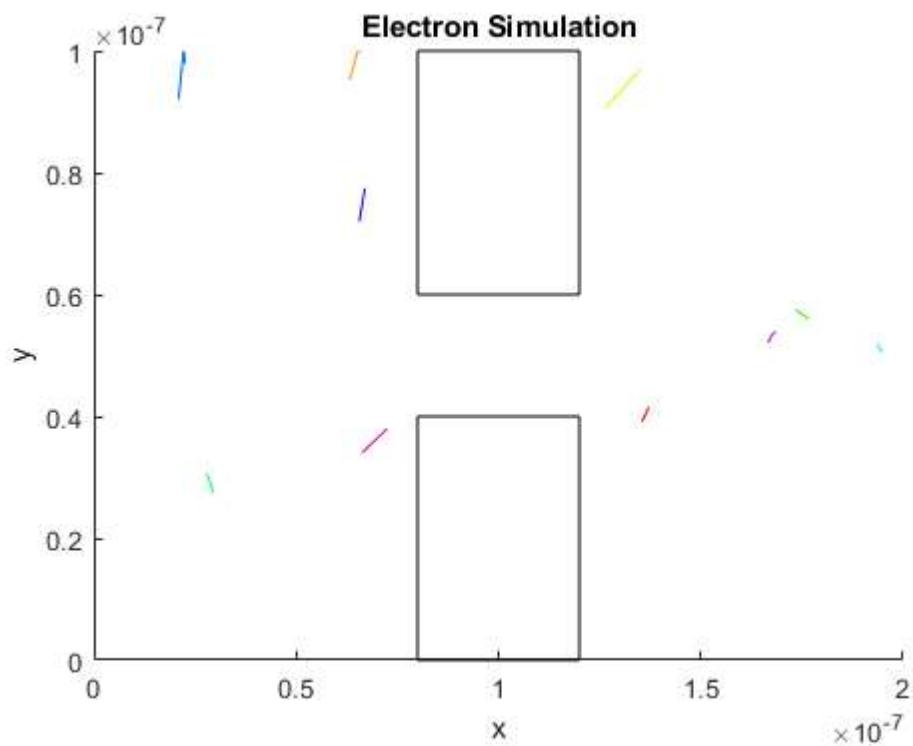
vy(ylt) = -(vy(ylt));
vy(ygt) = -(vy(ygt));
```

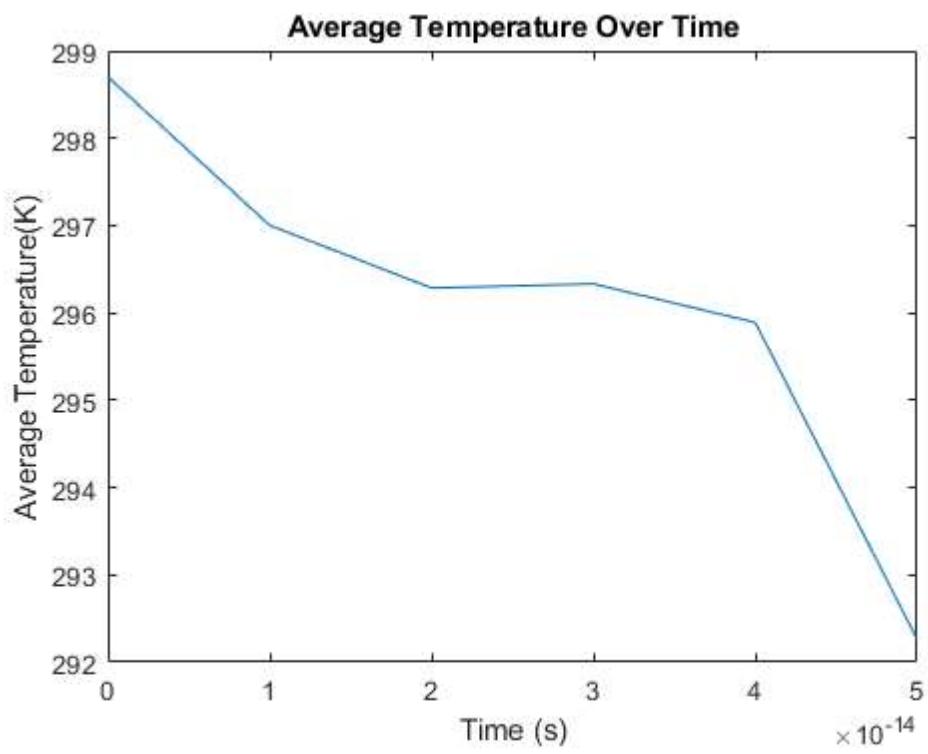
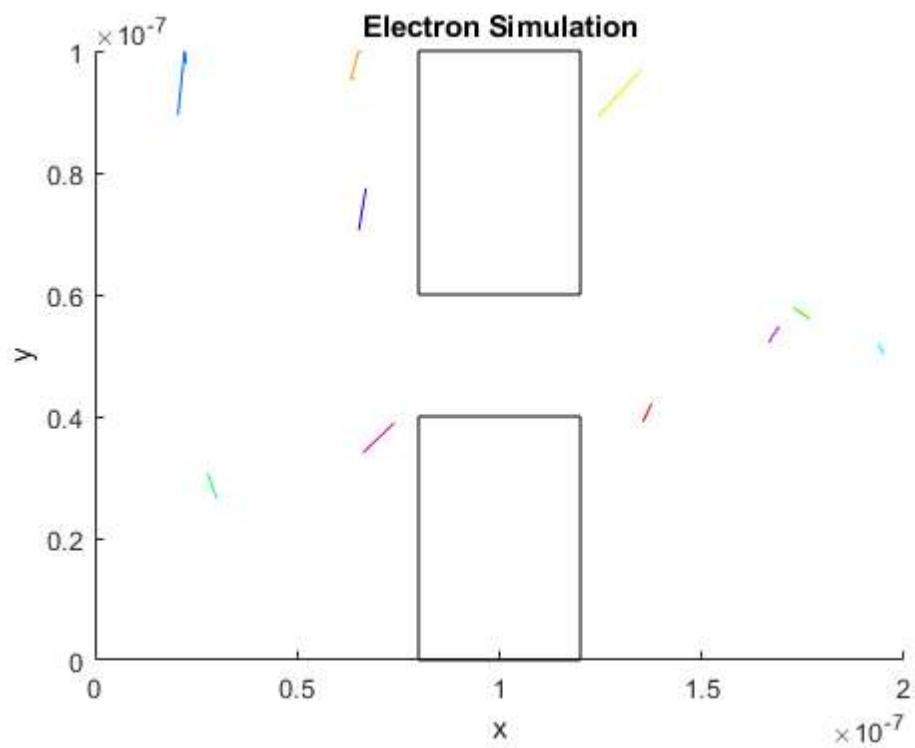



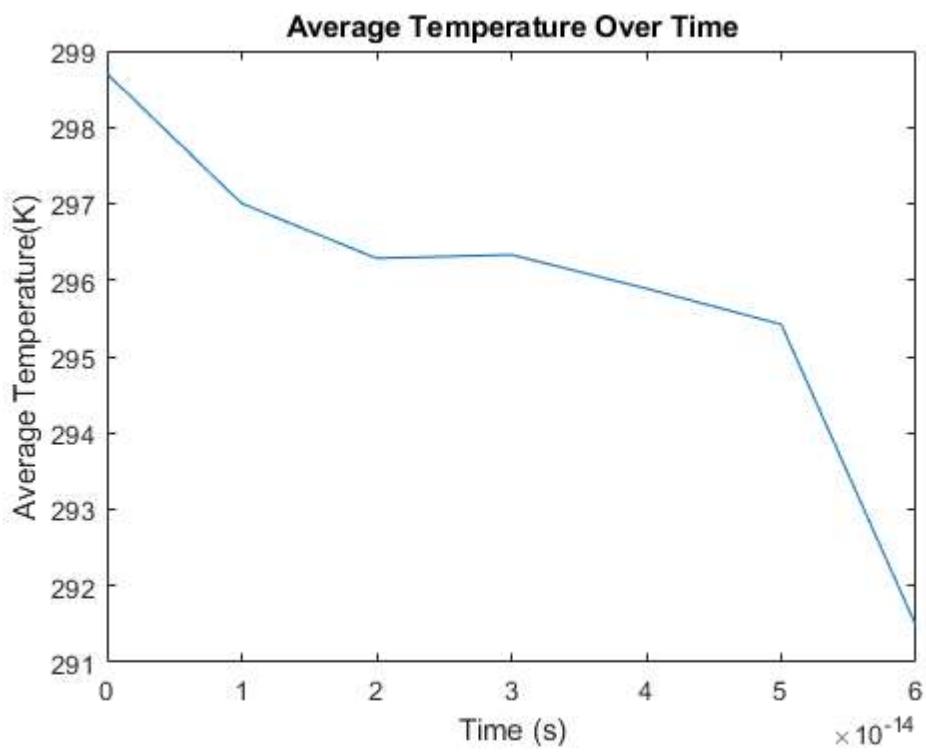
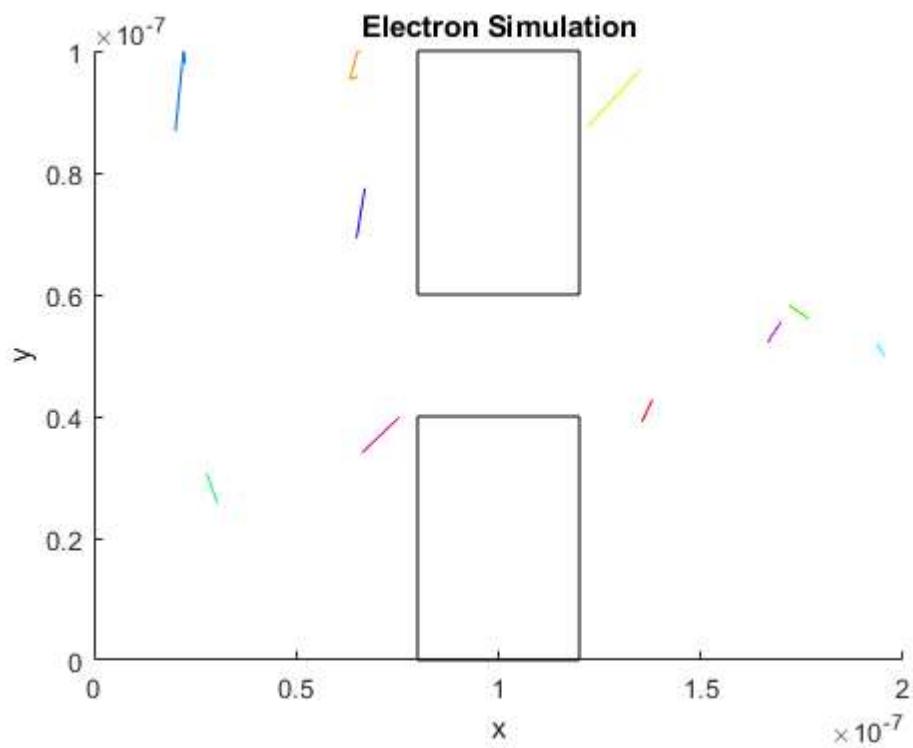
$\times 10^{-7}$ **Electron Simulation**

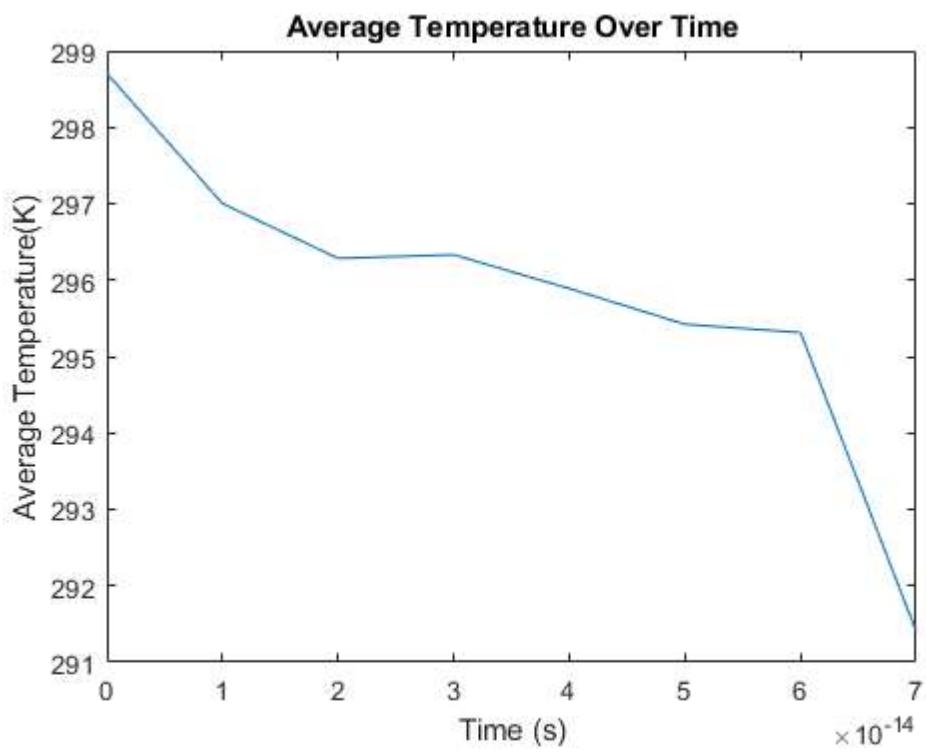
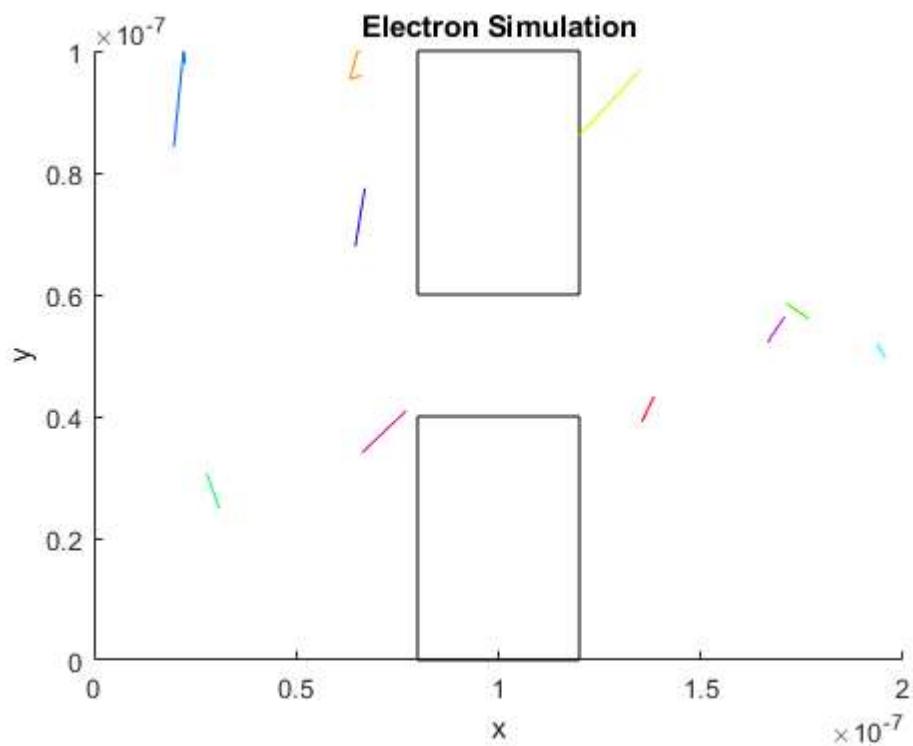


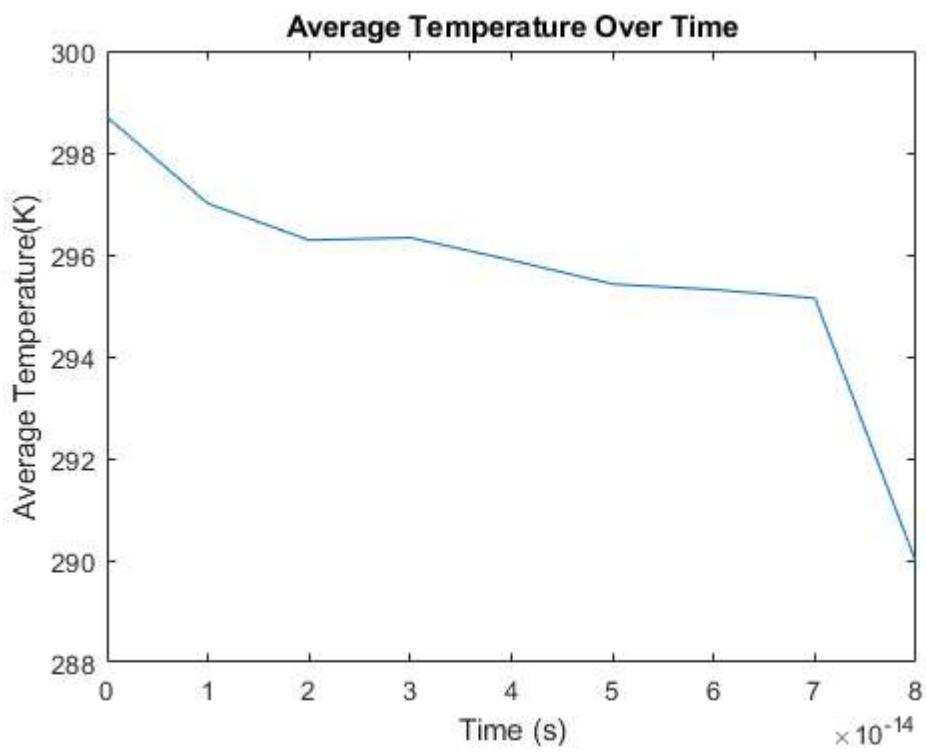
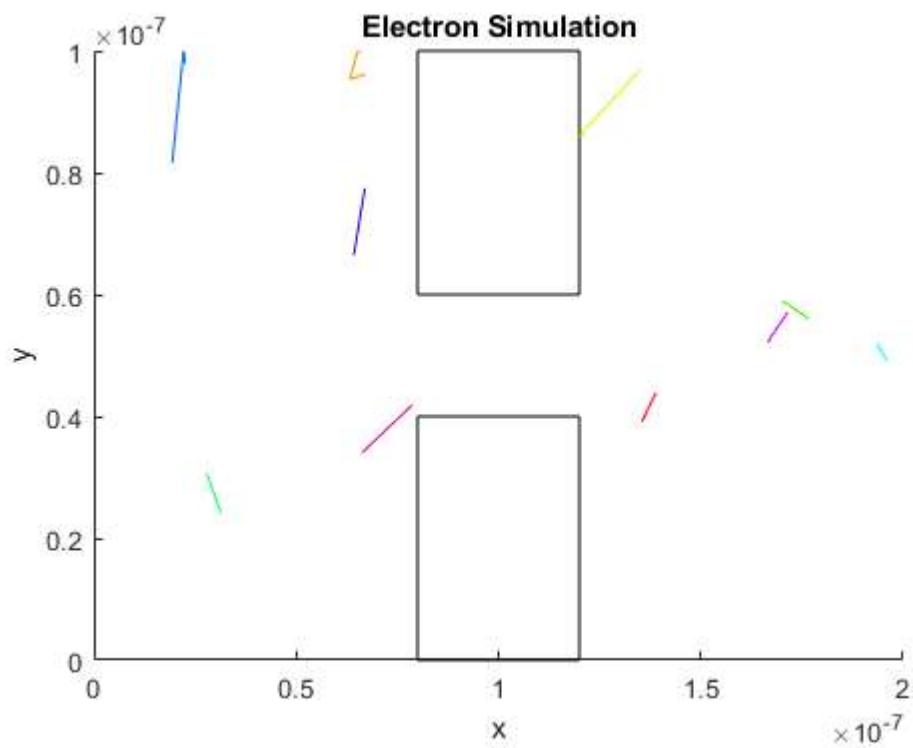


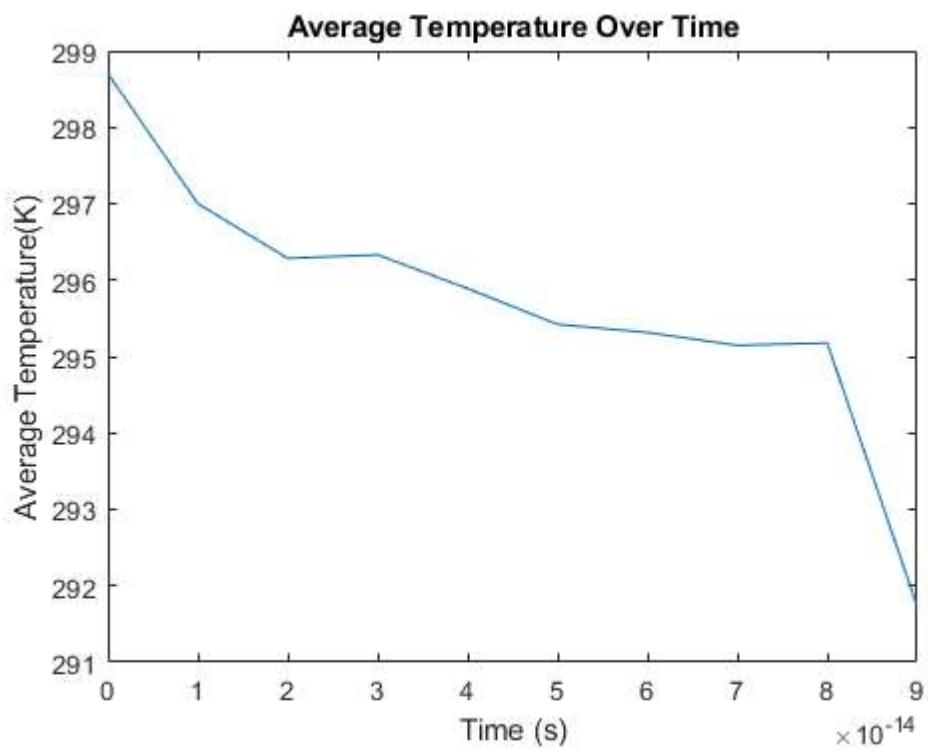
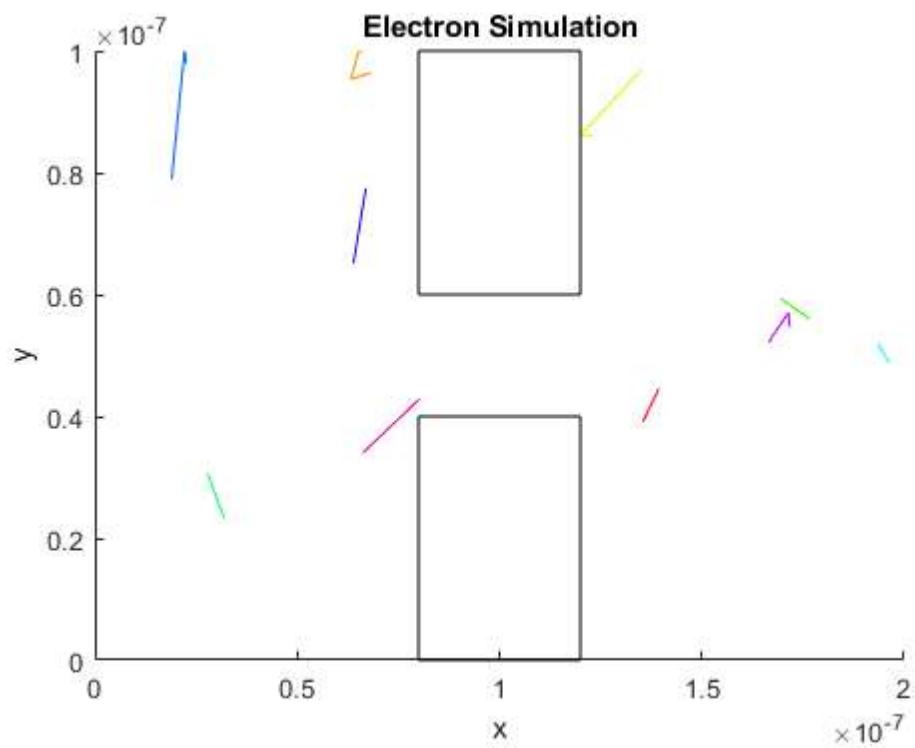


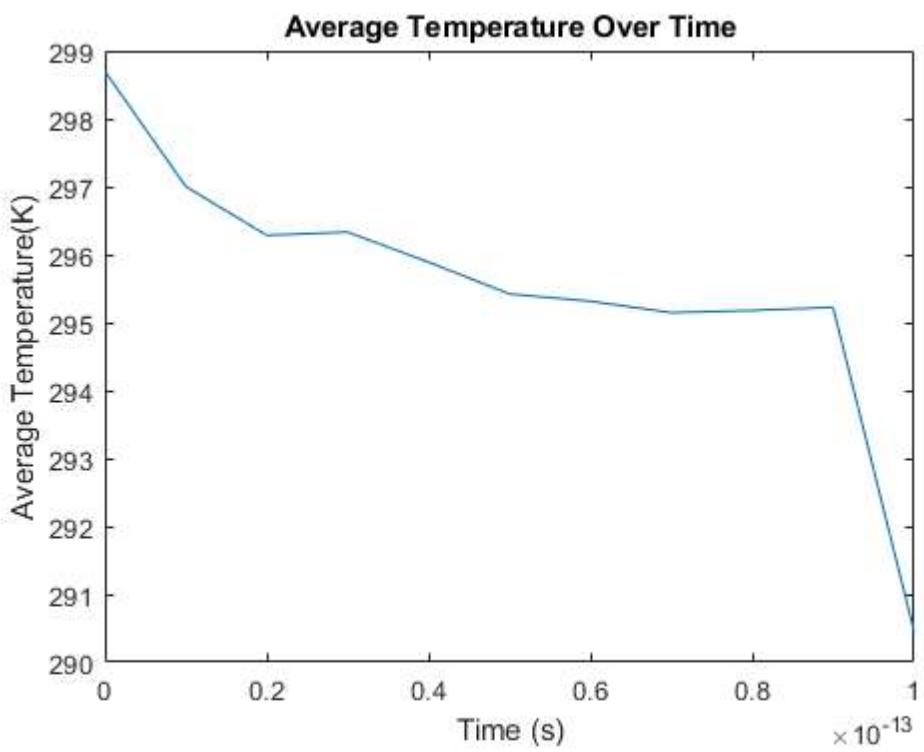
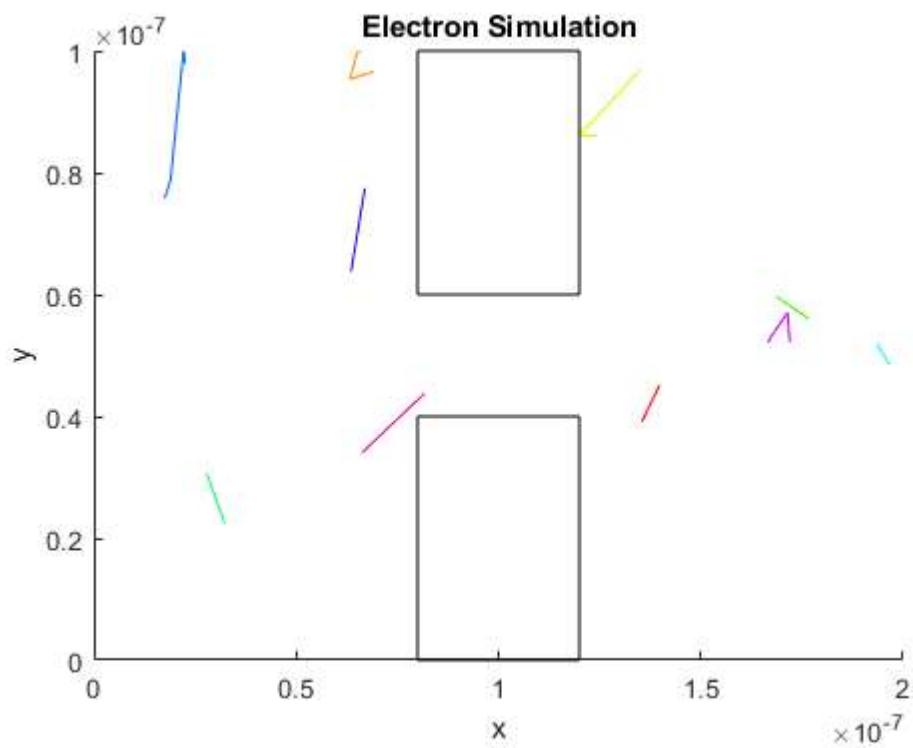


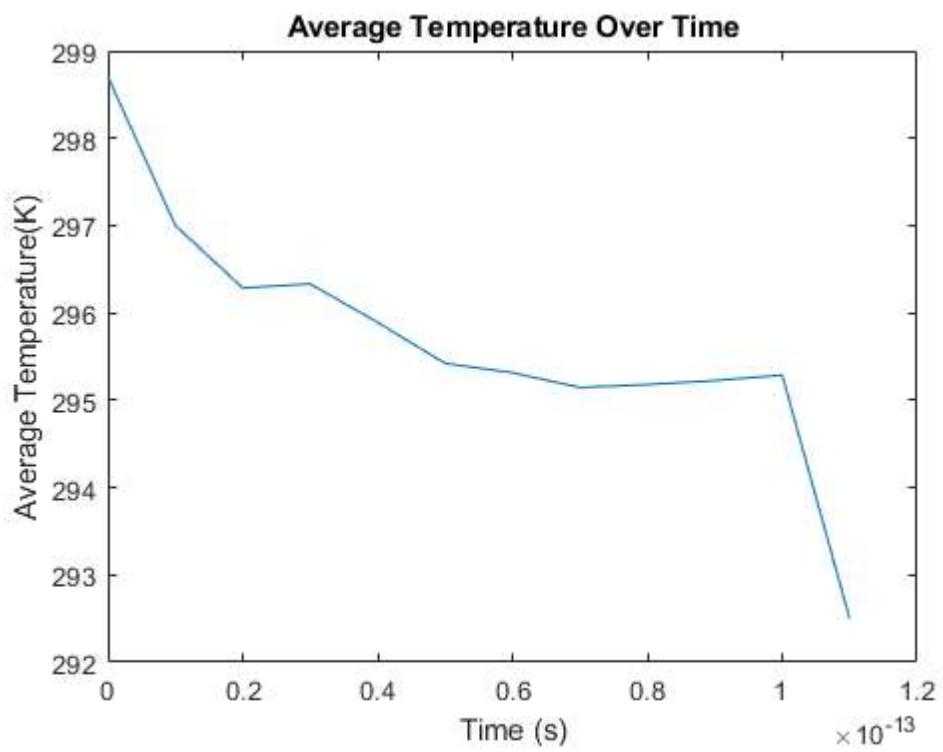
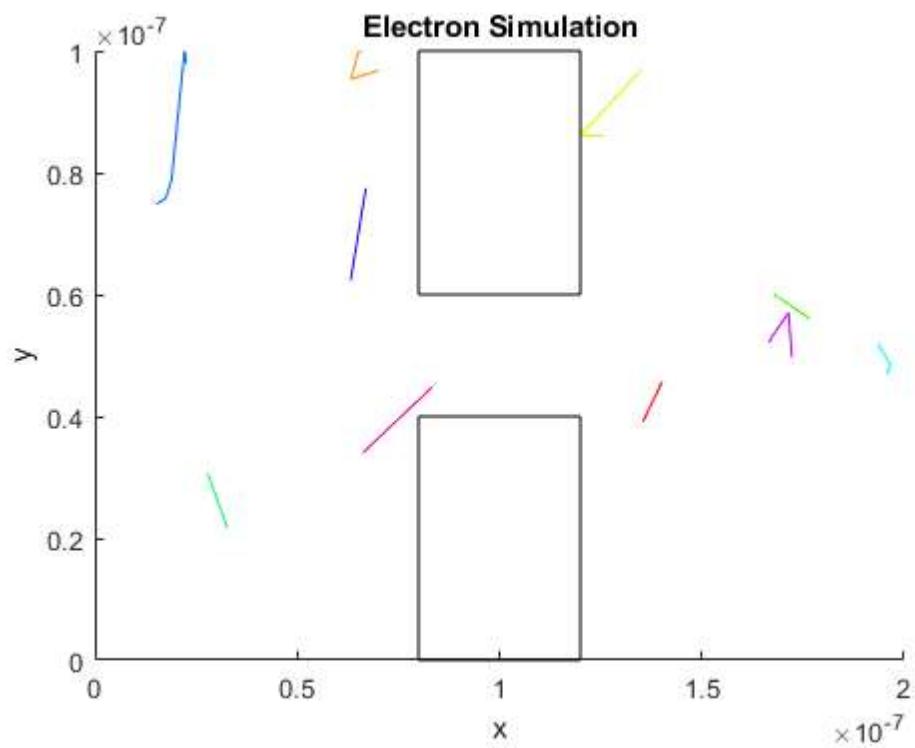


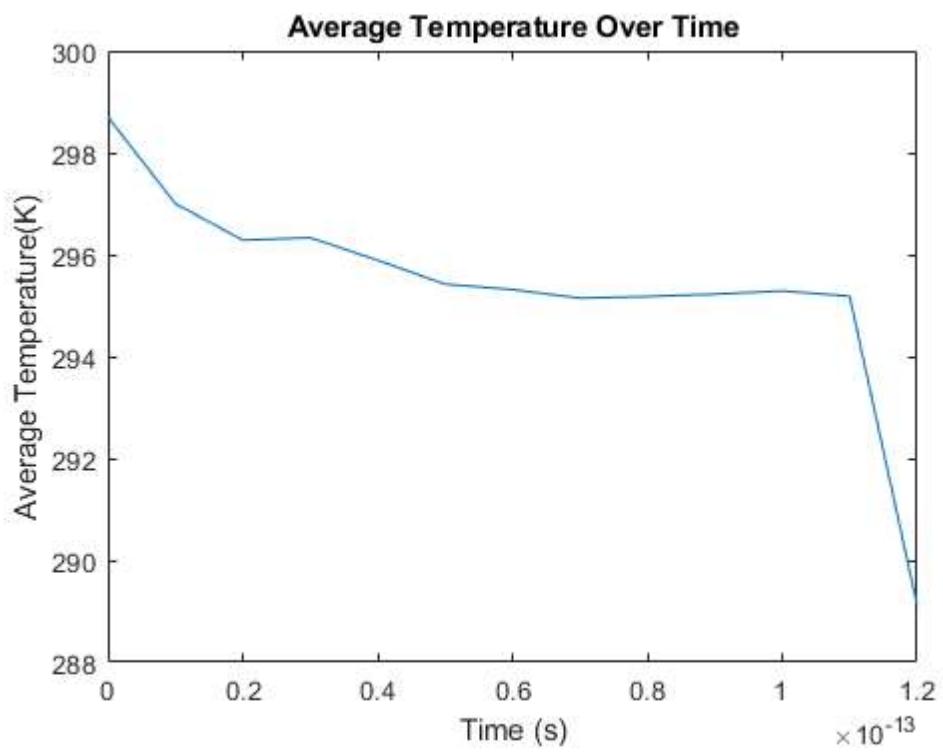
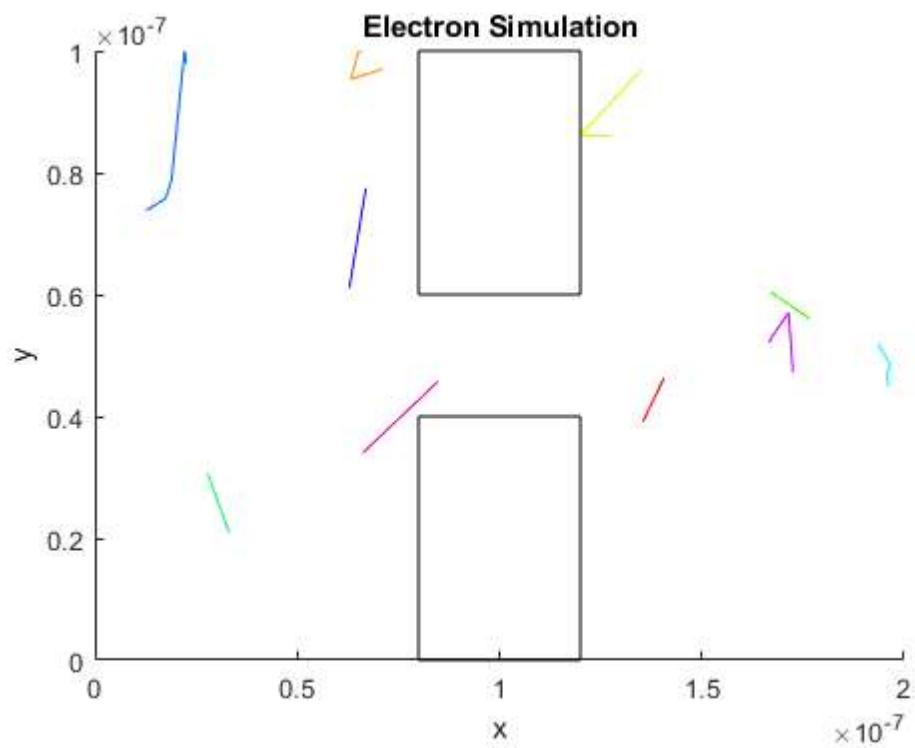


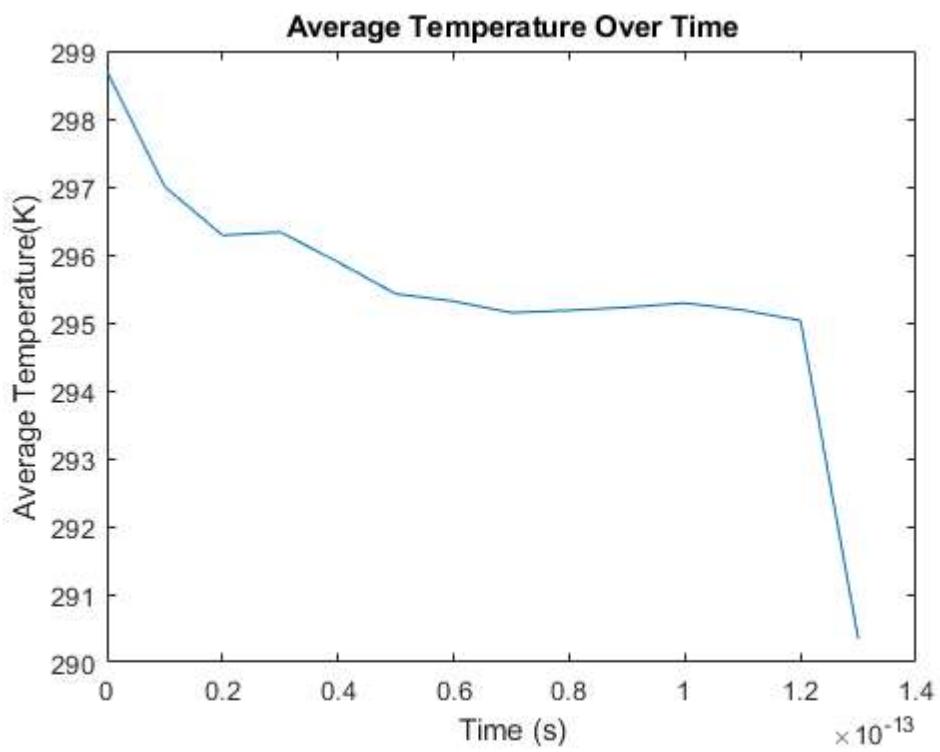
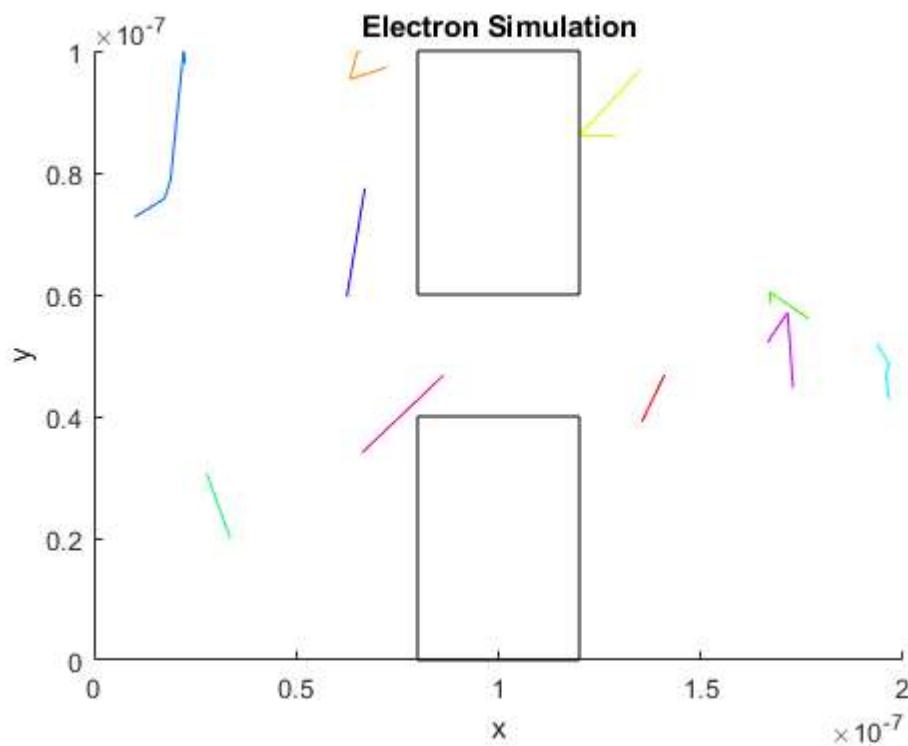


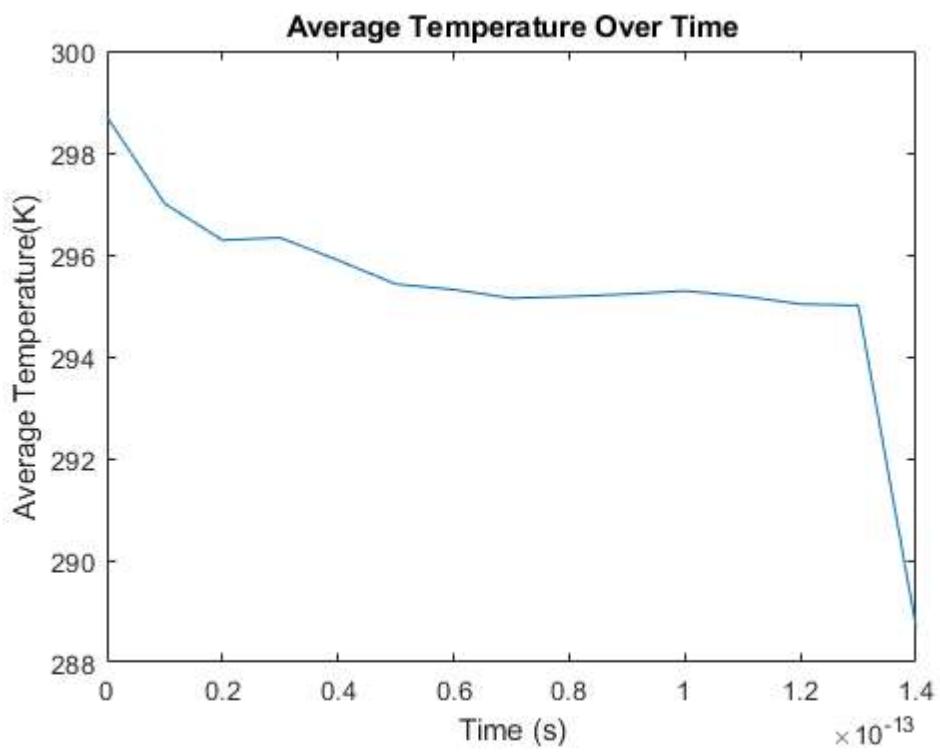
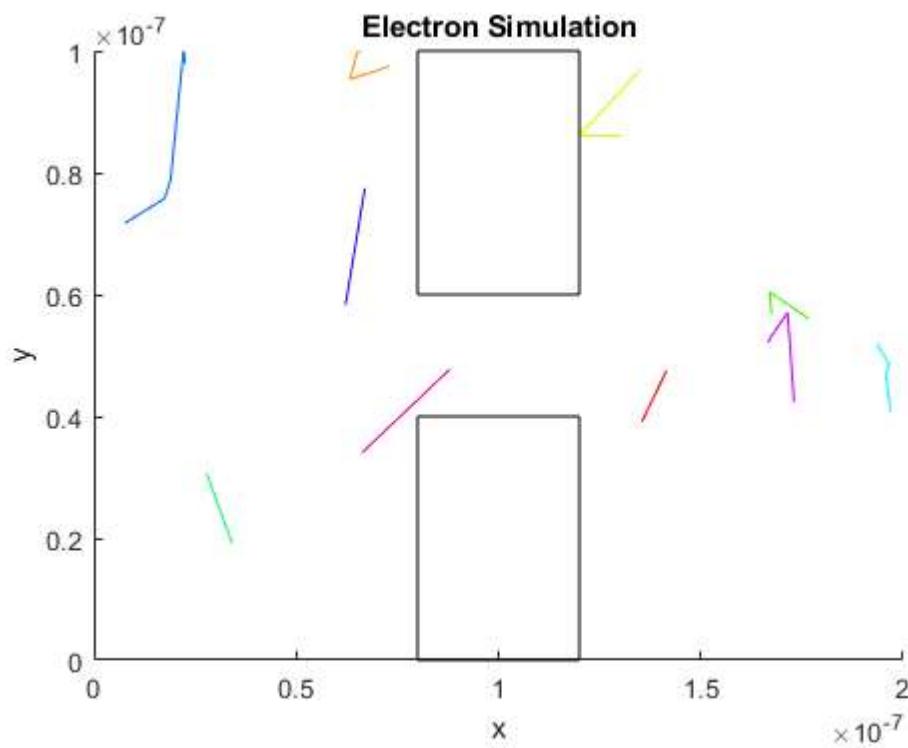


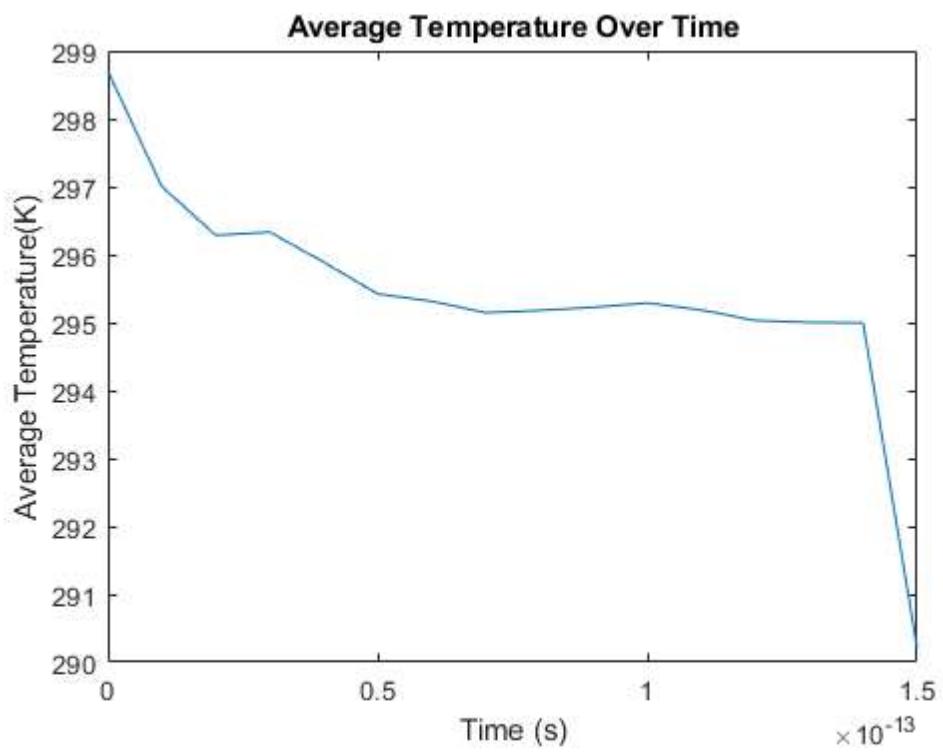
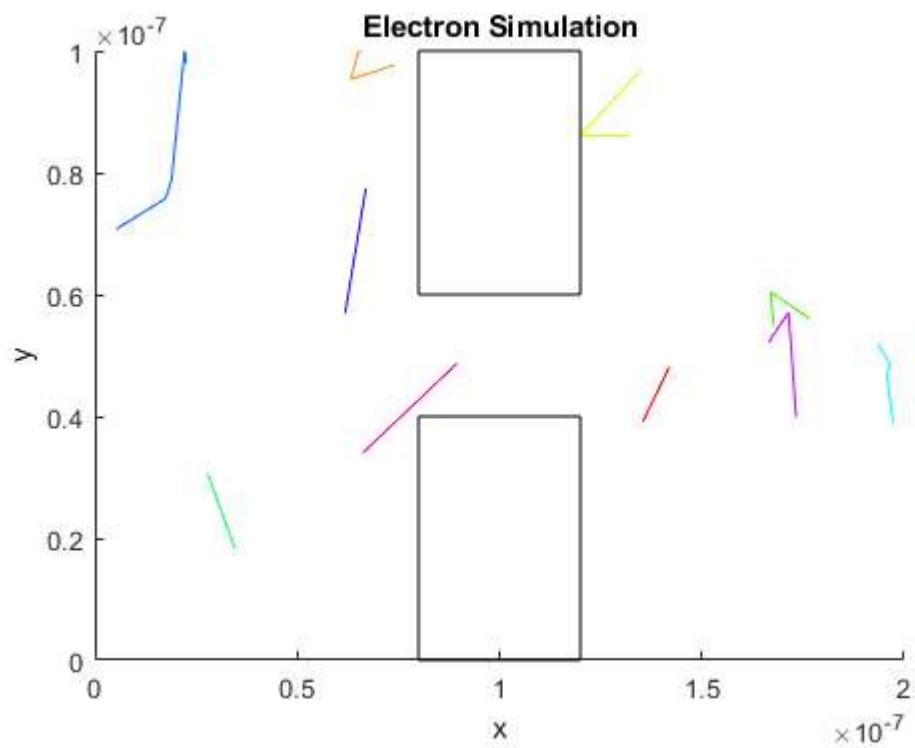


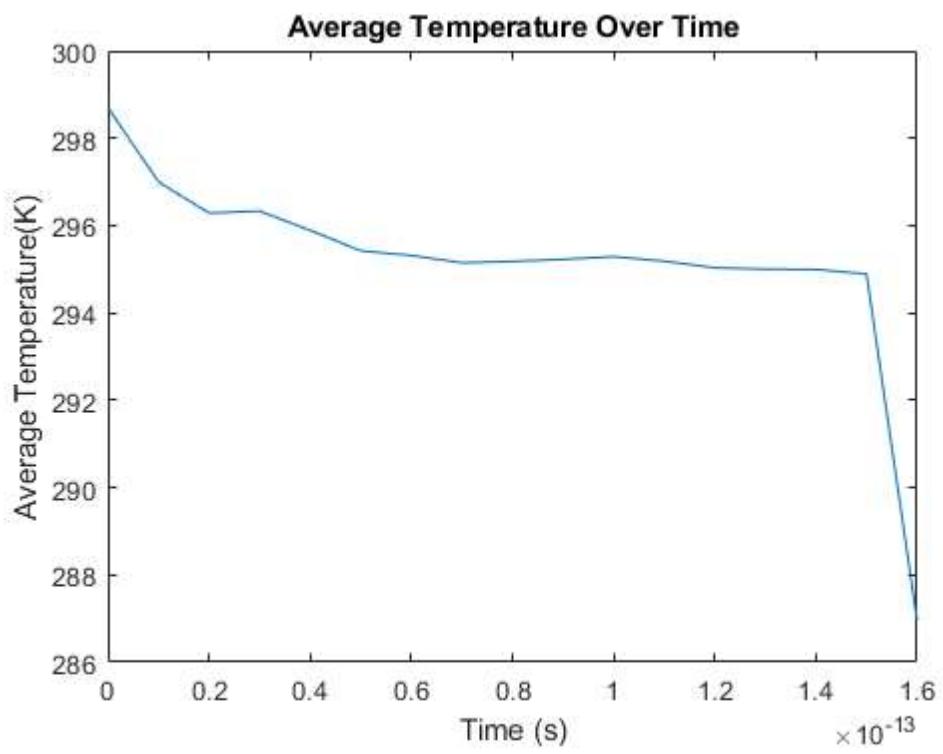
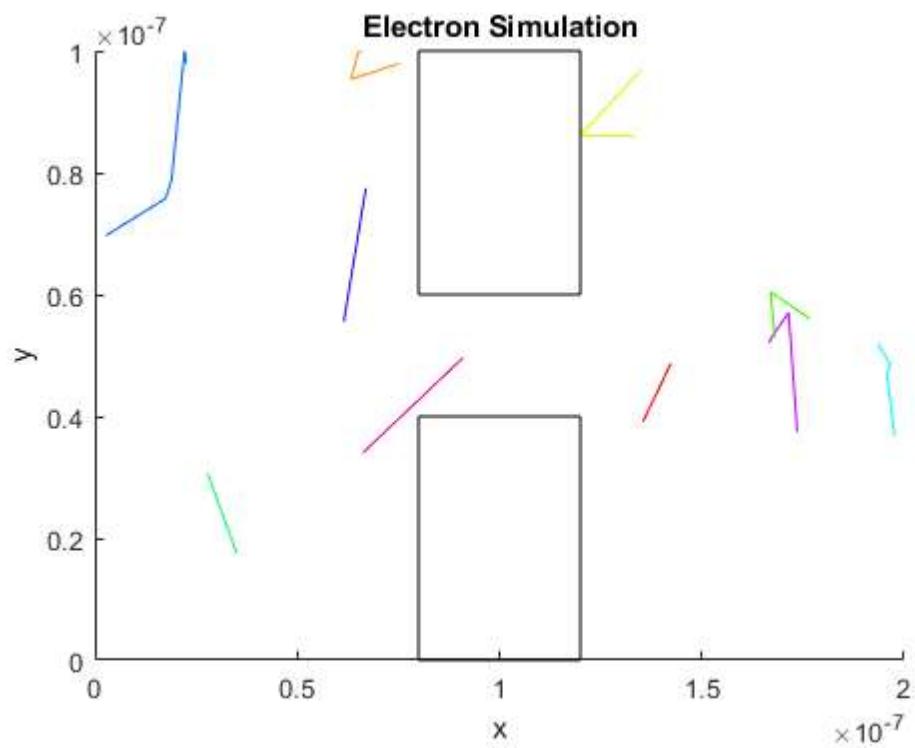


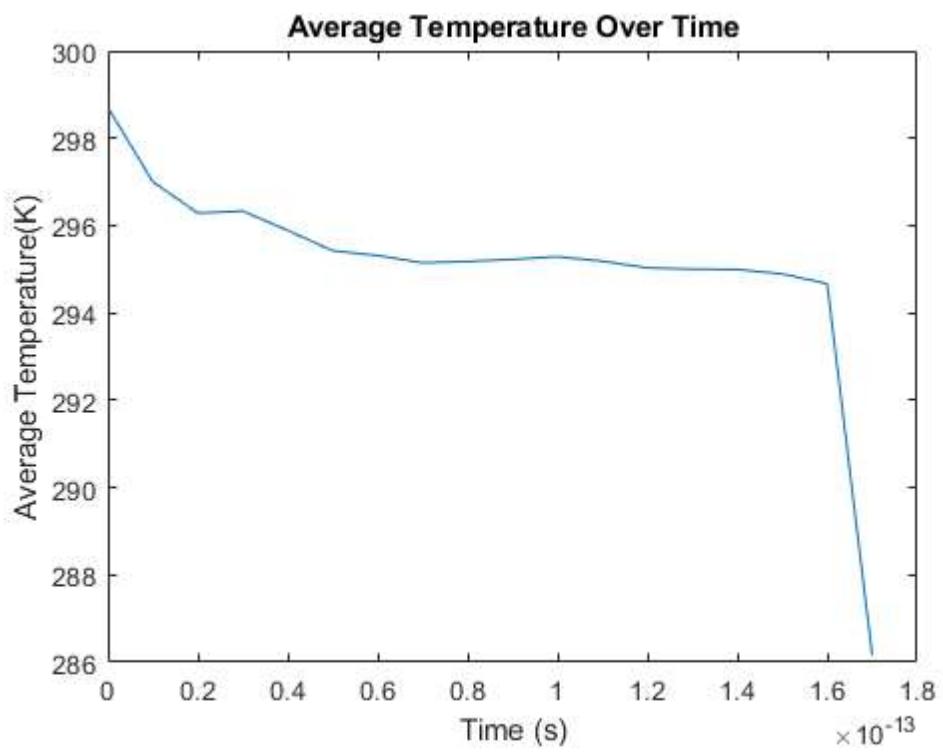
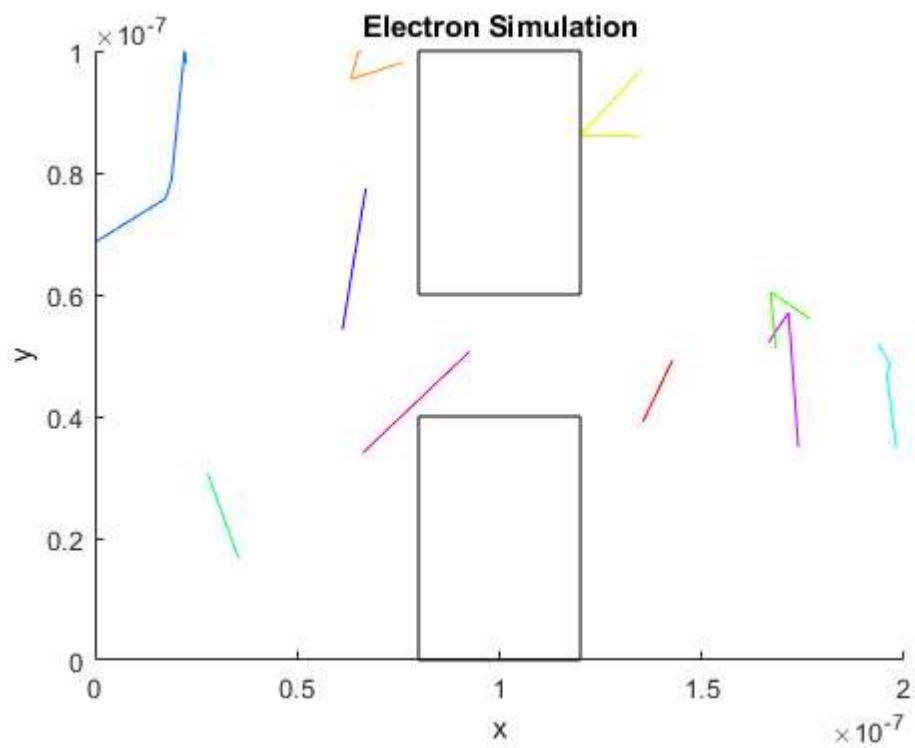


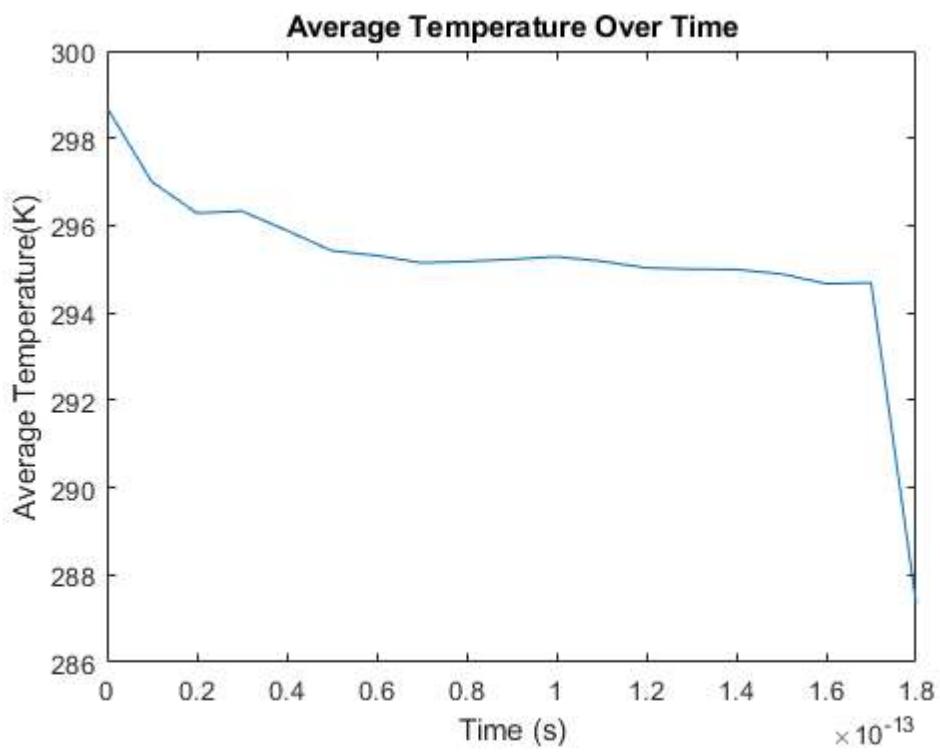
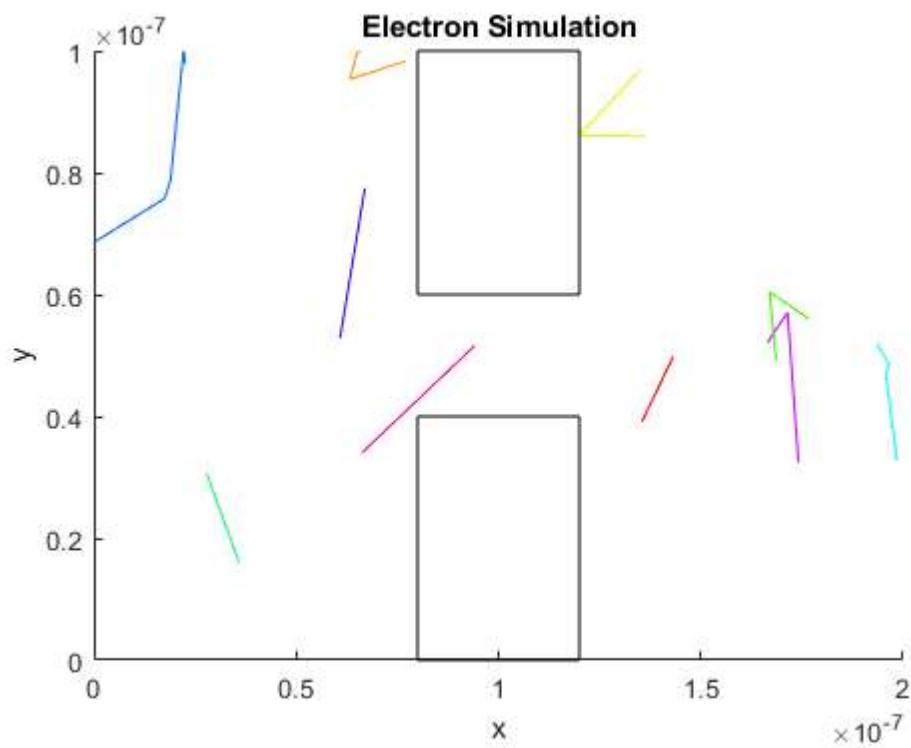


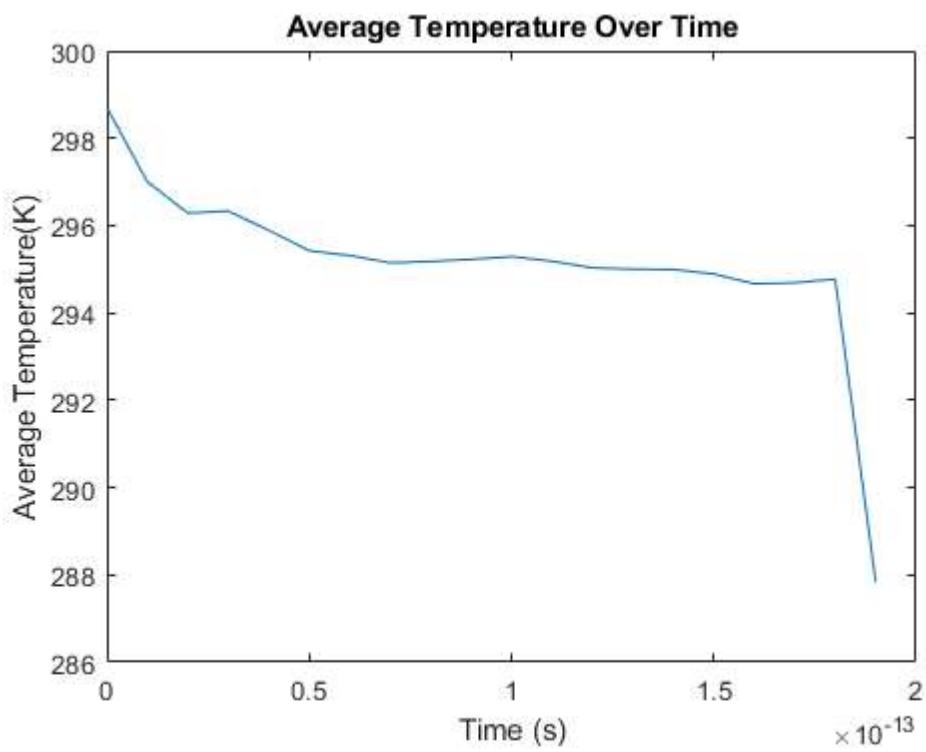
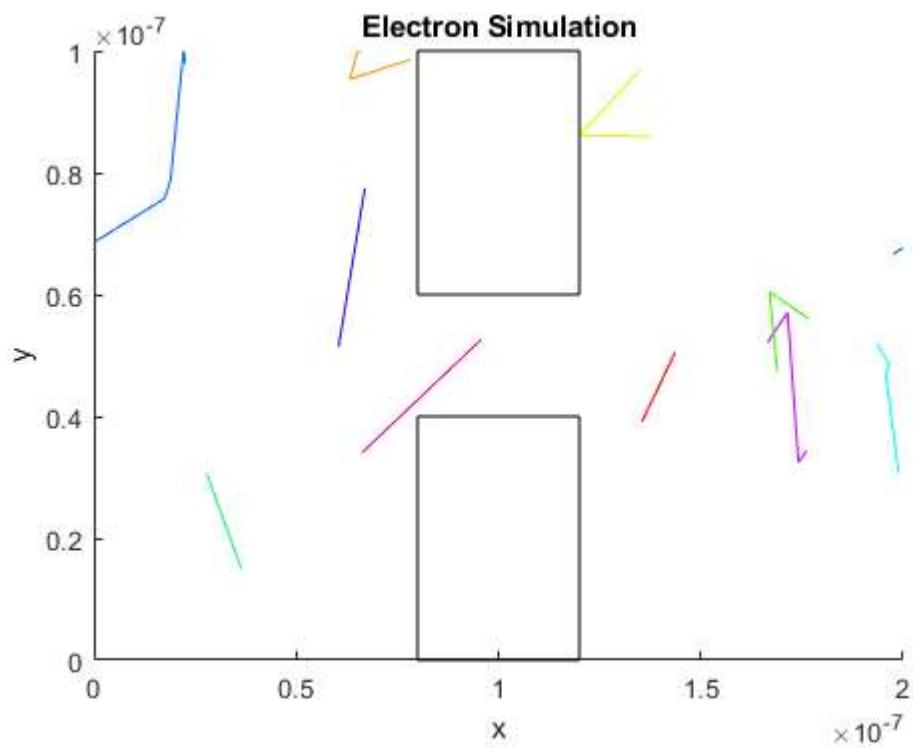


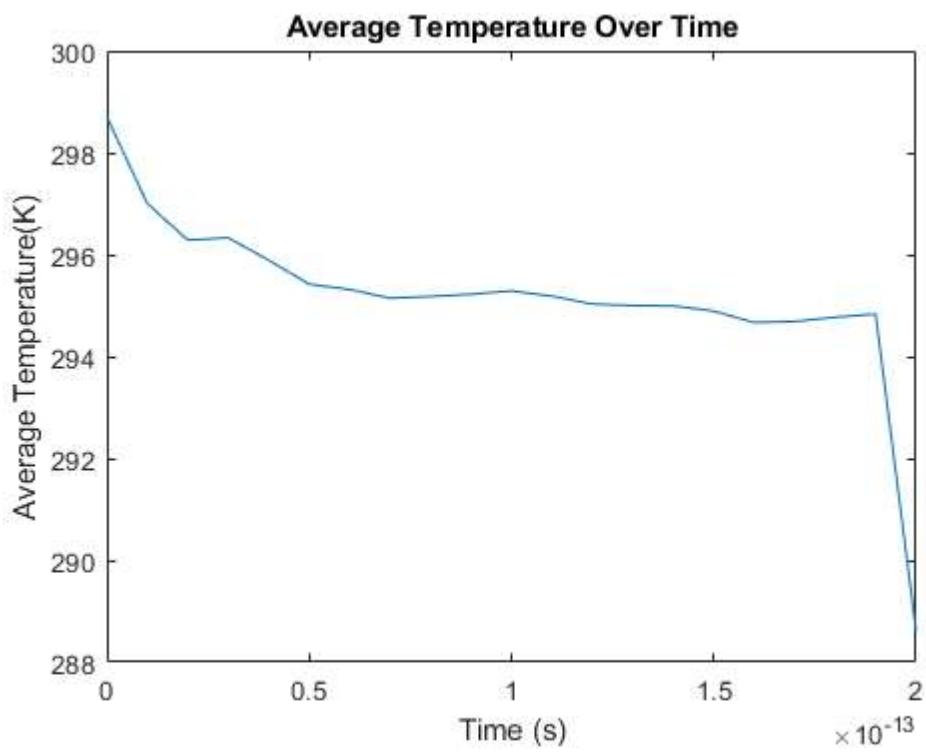
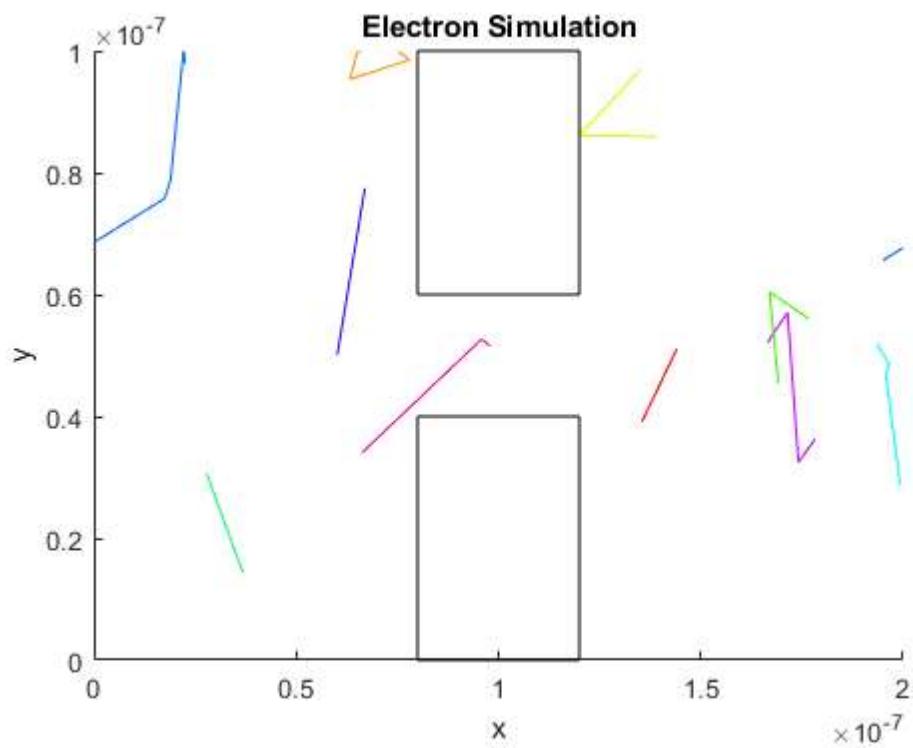


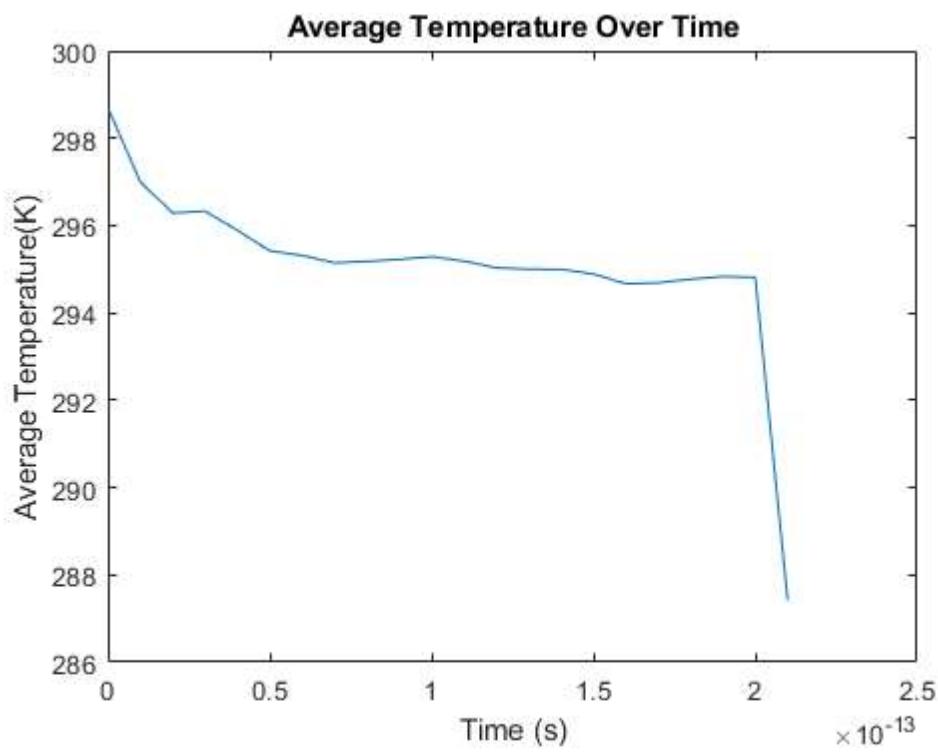
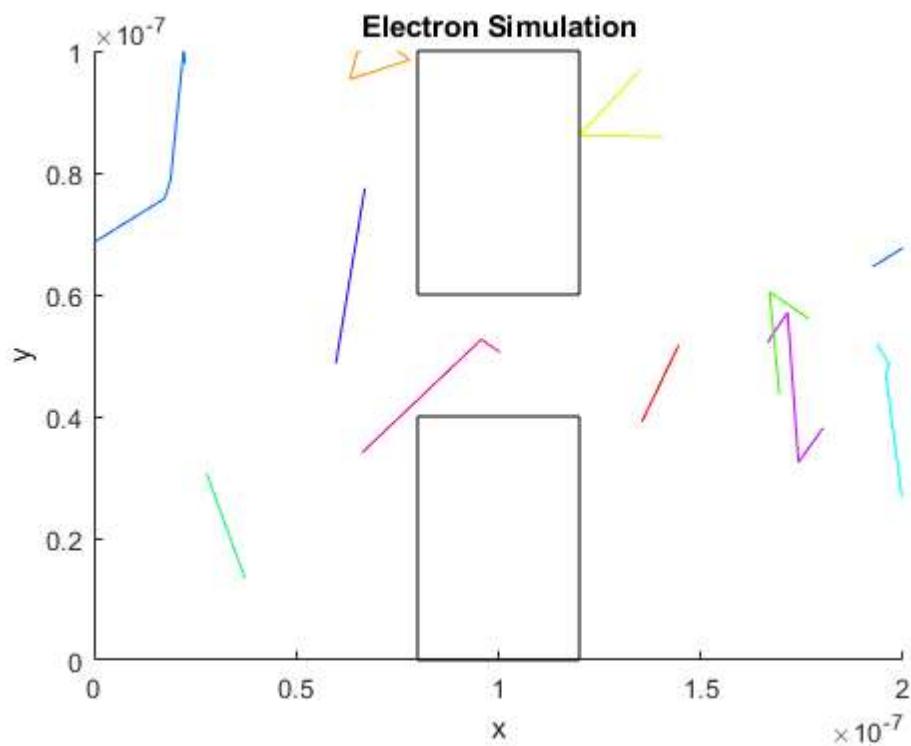


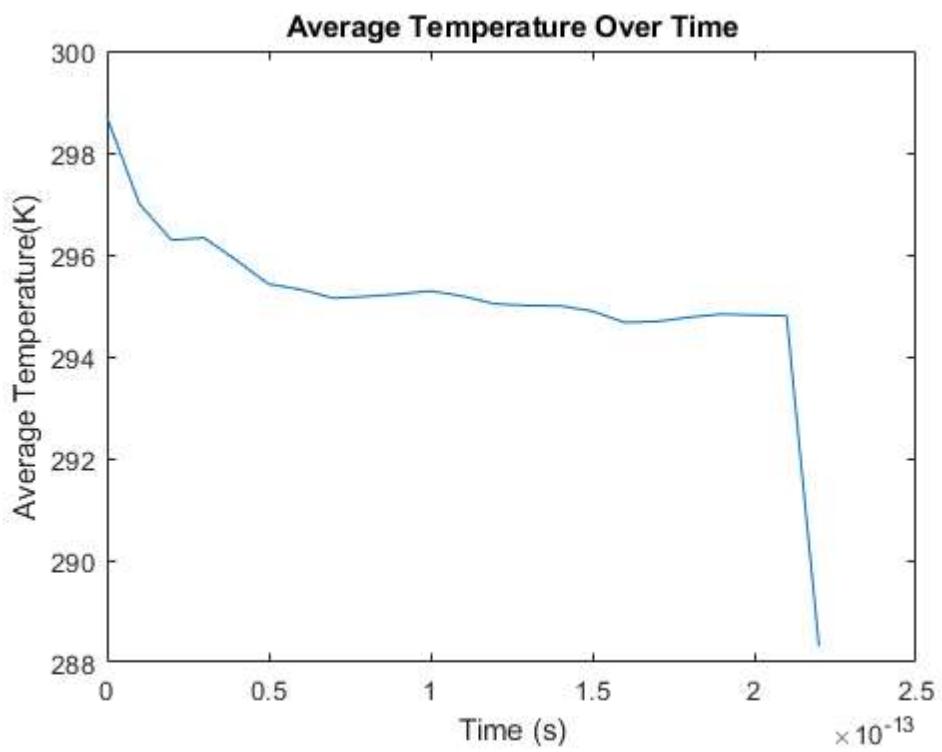
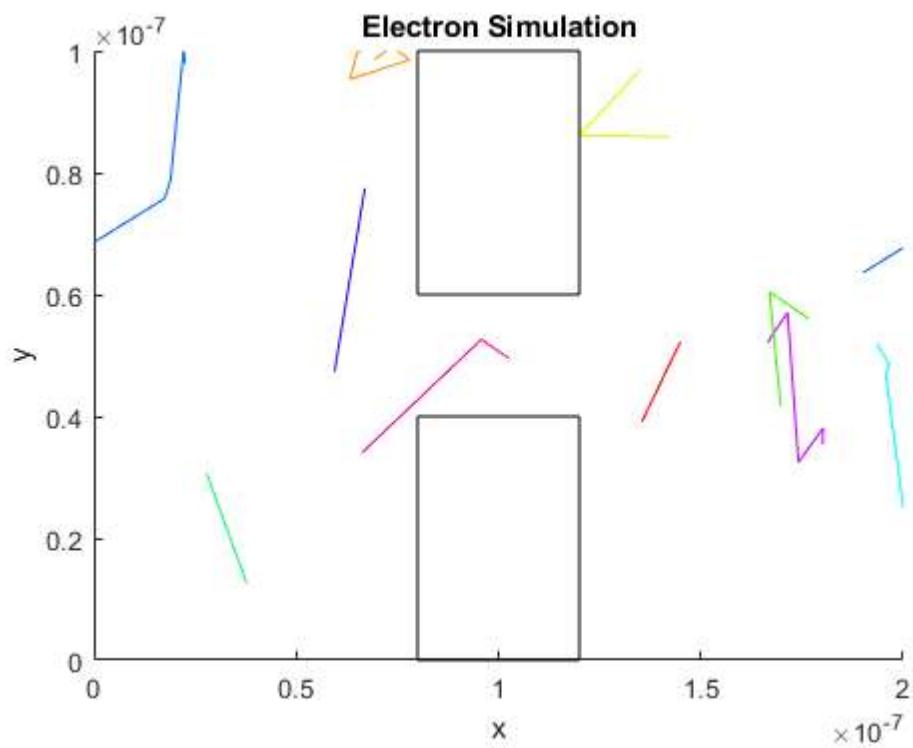


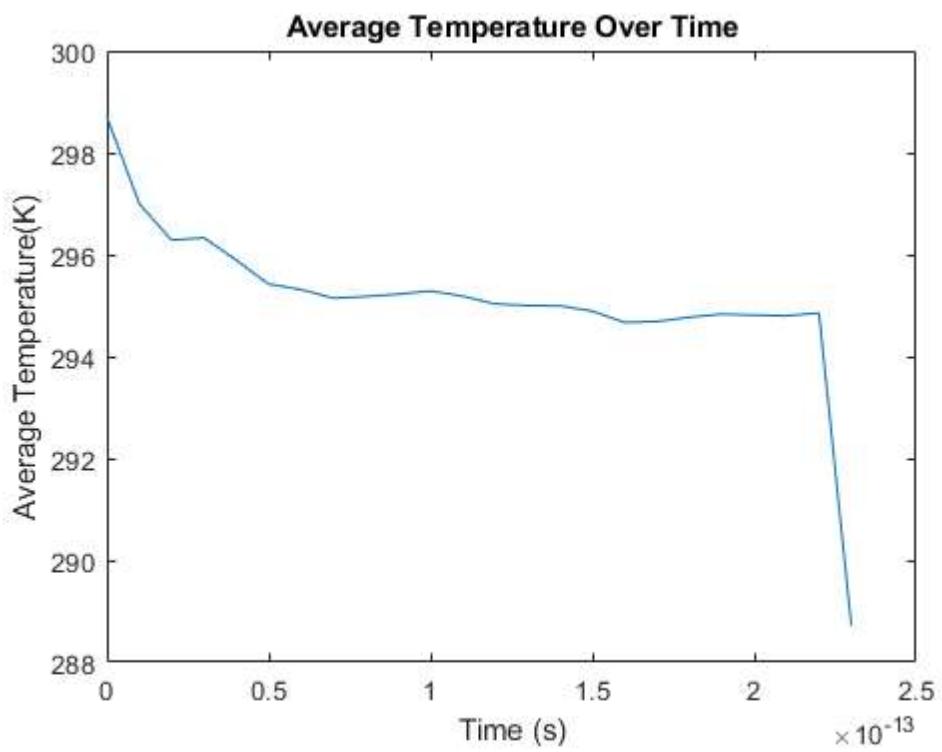
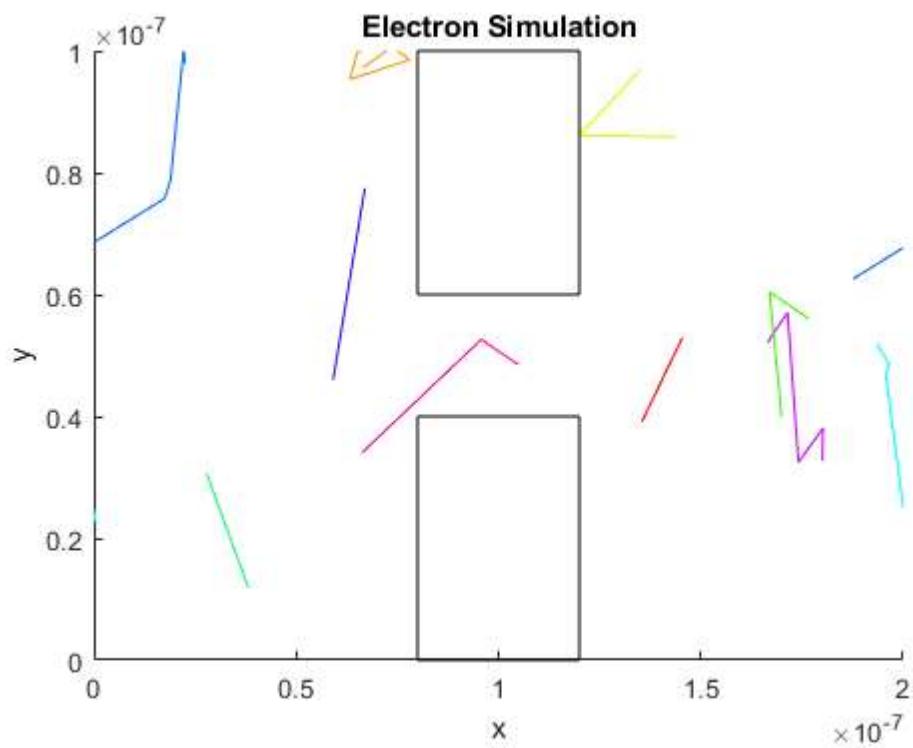


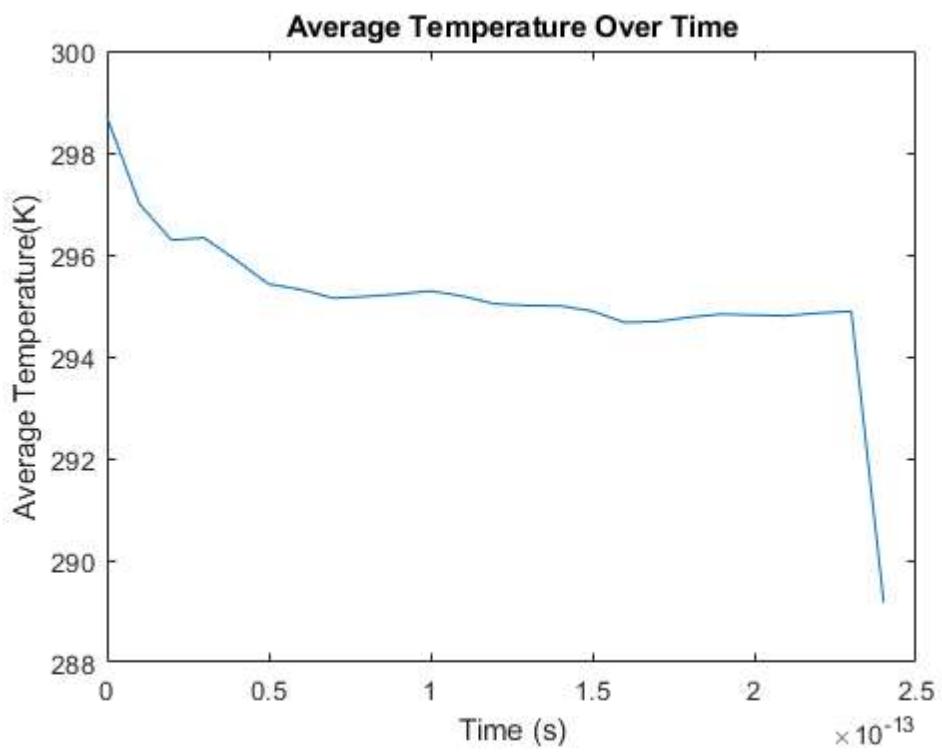
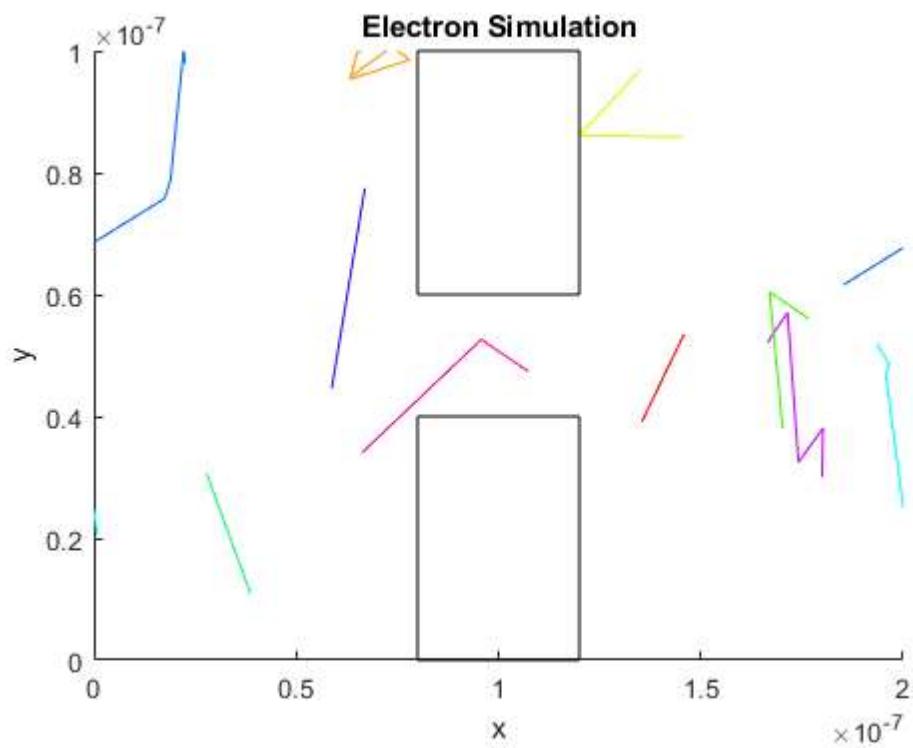


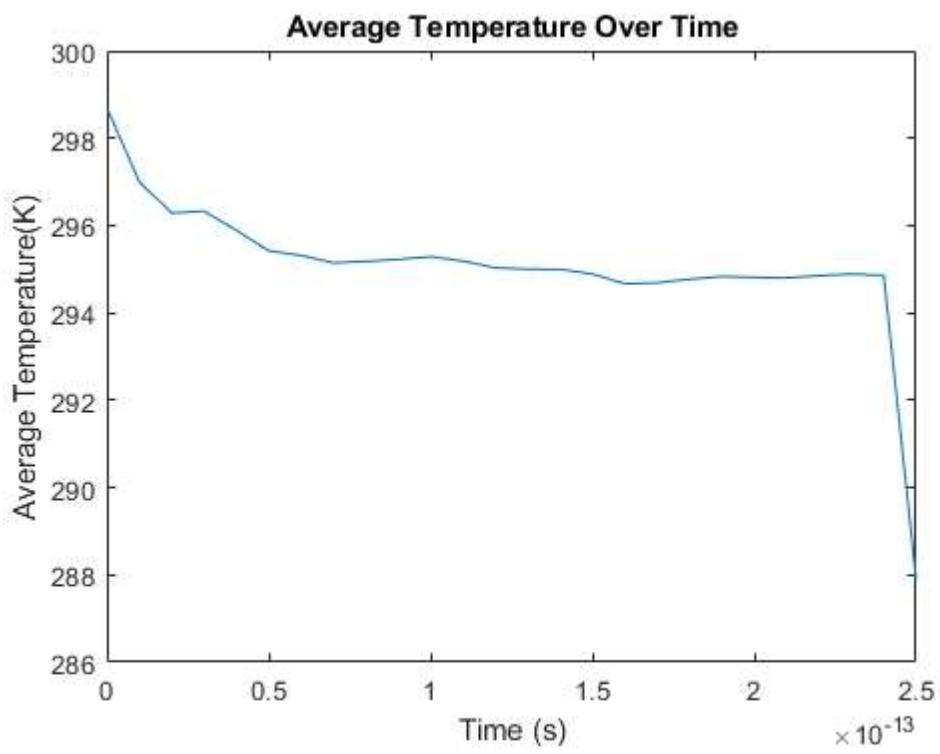
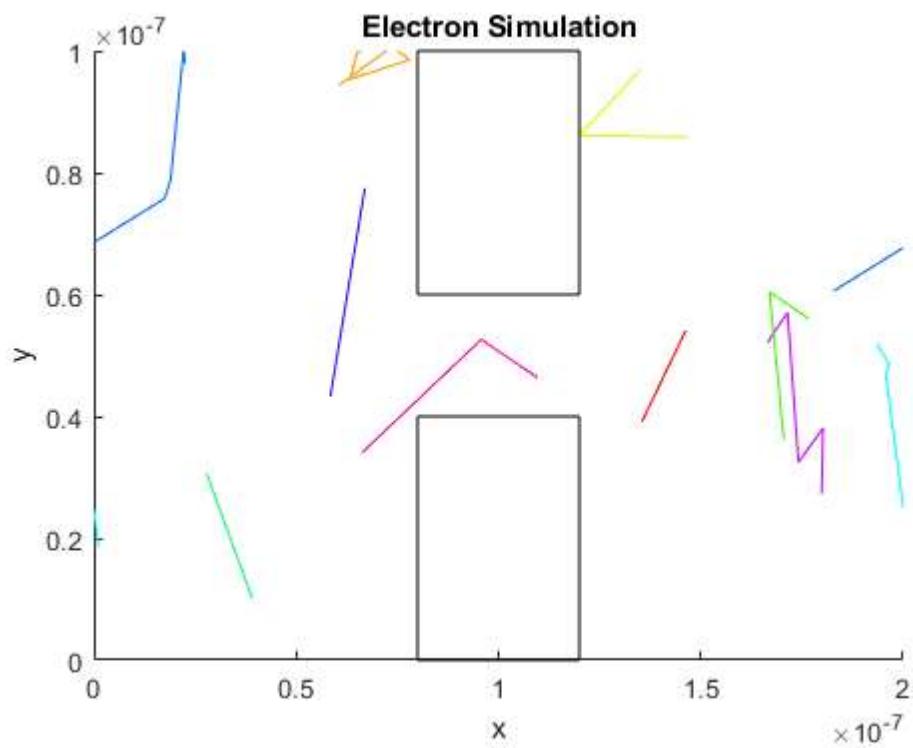


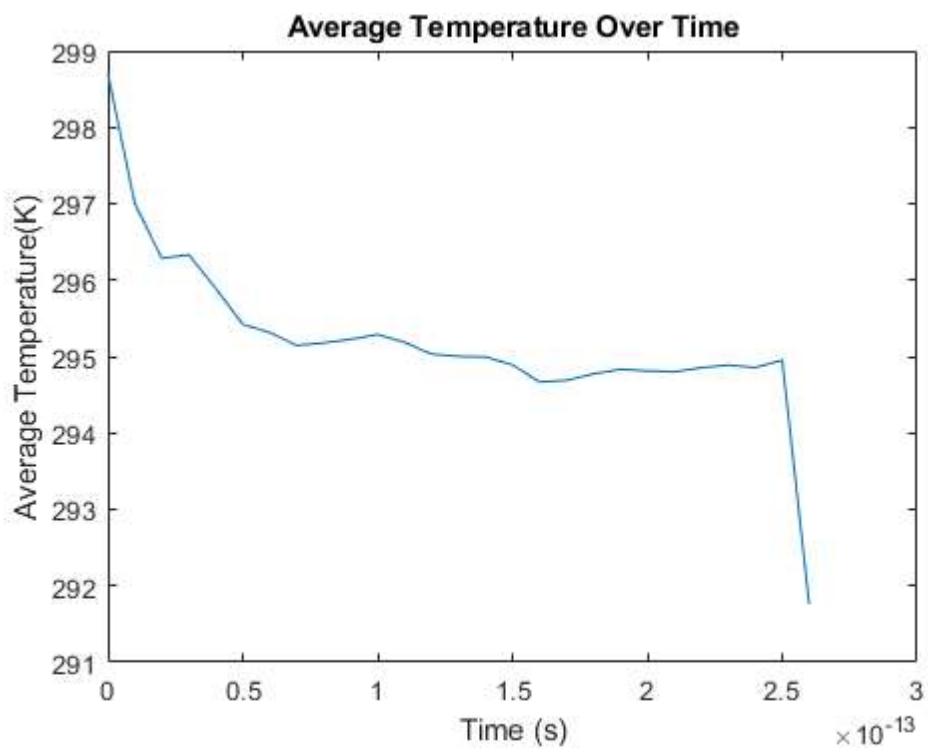
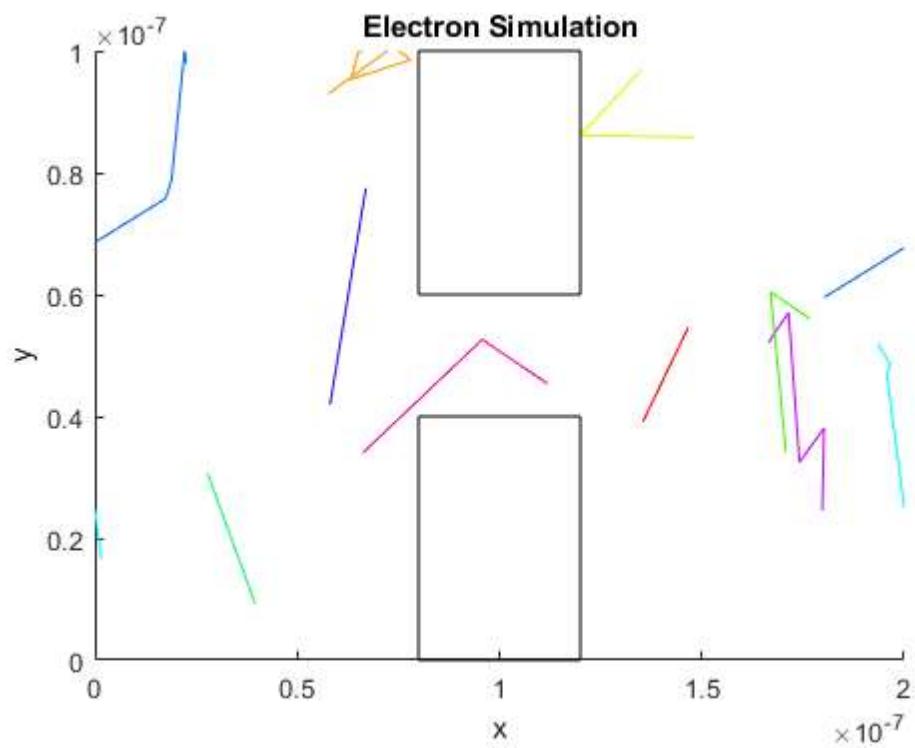


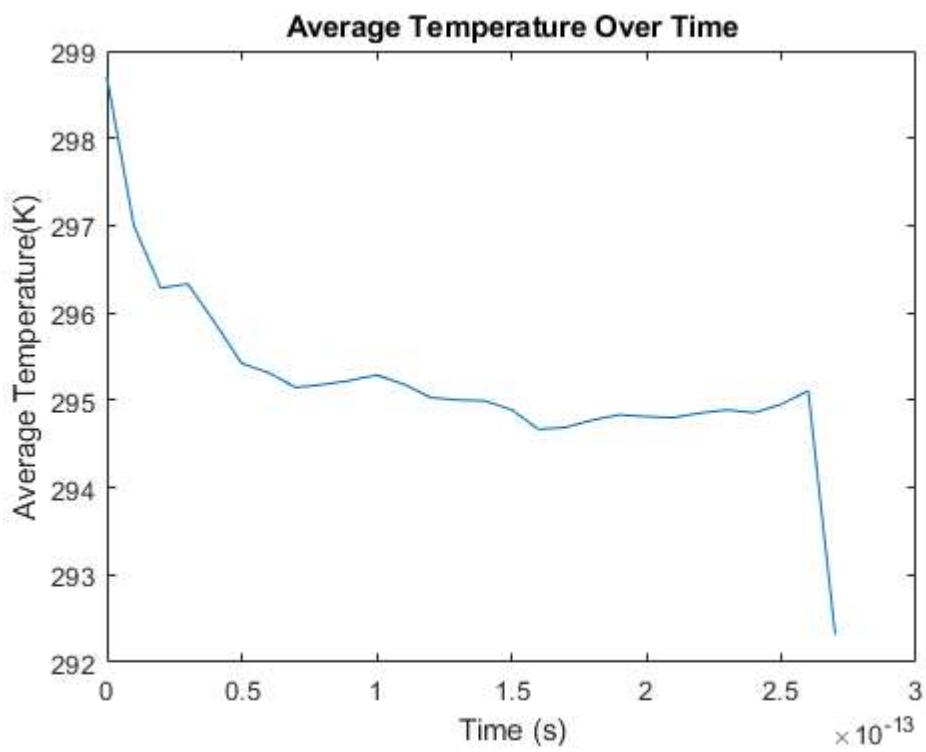
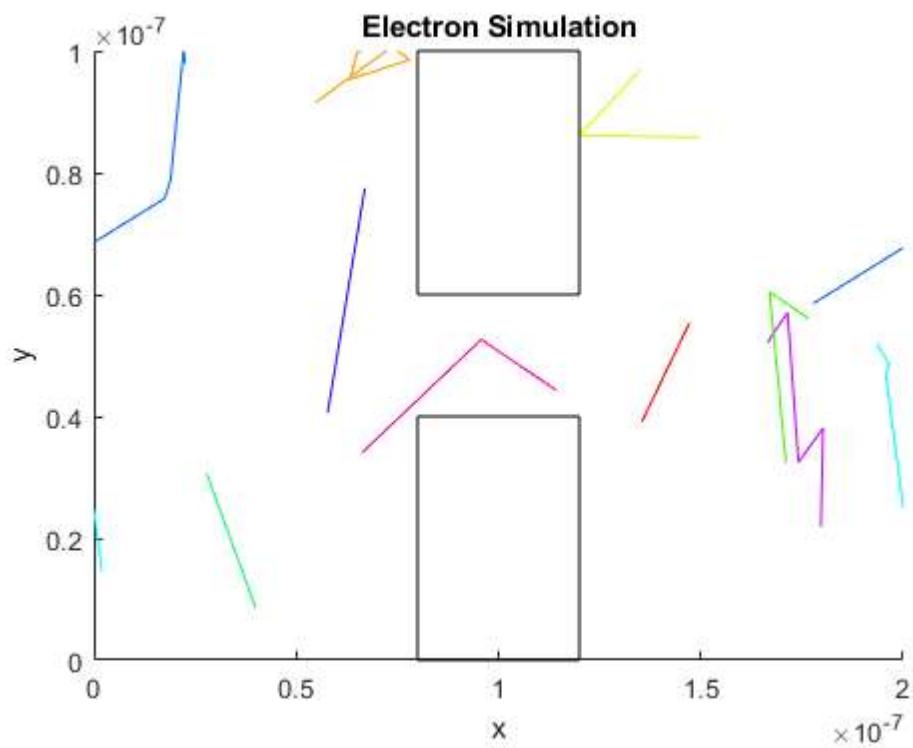


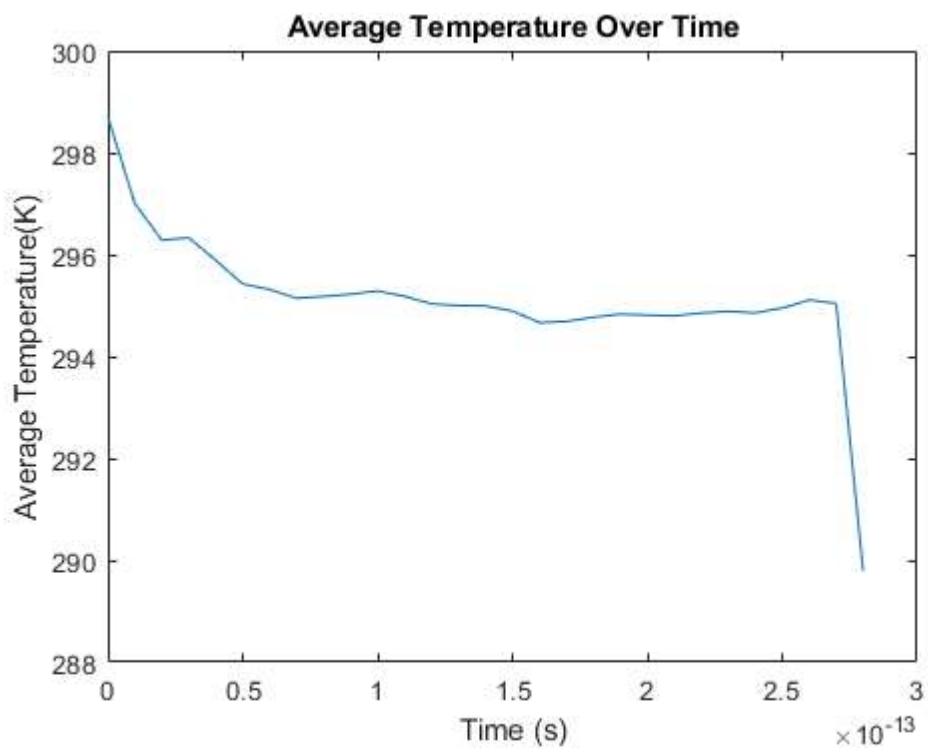
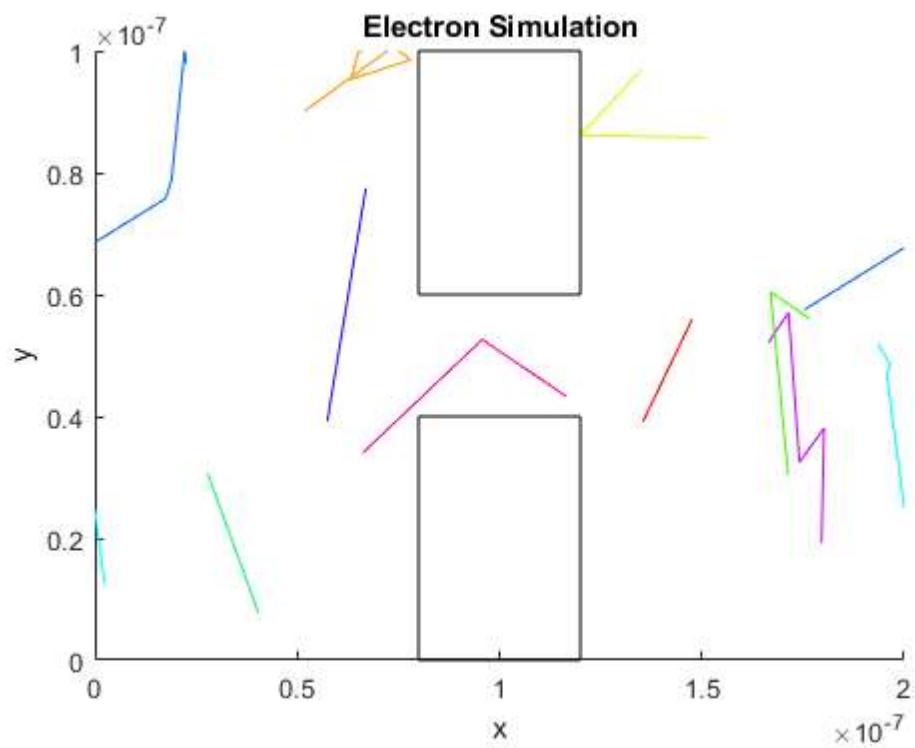


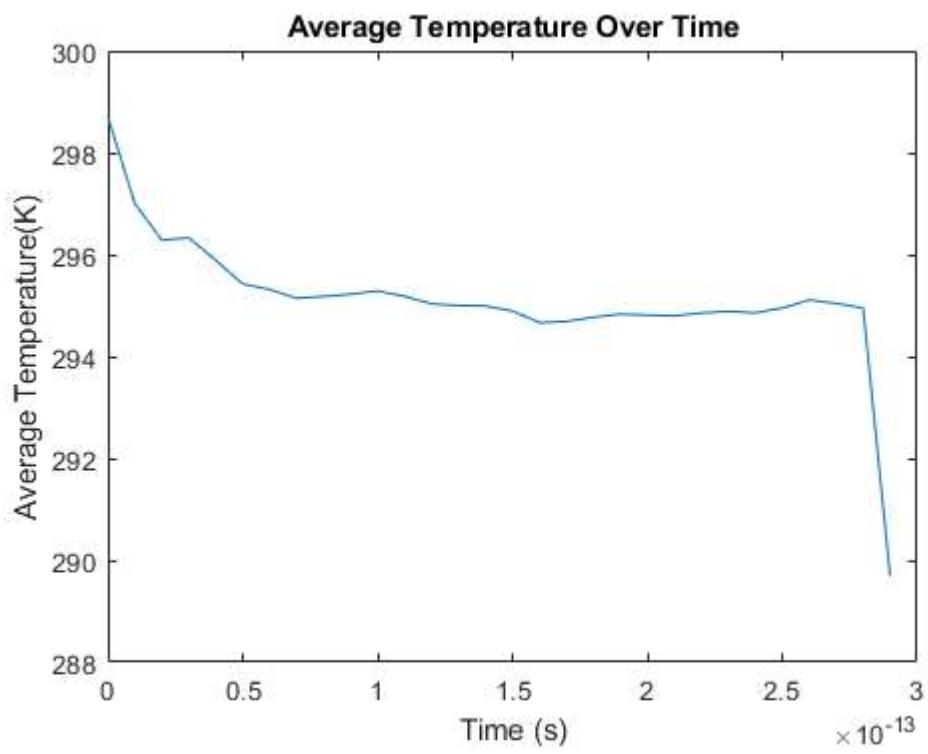
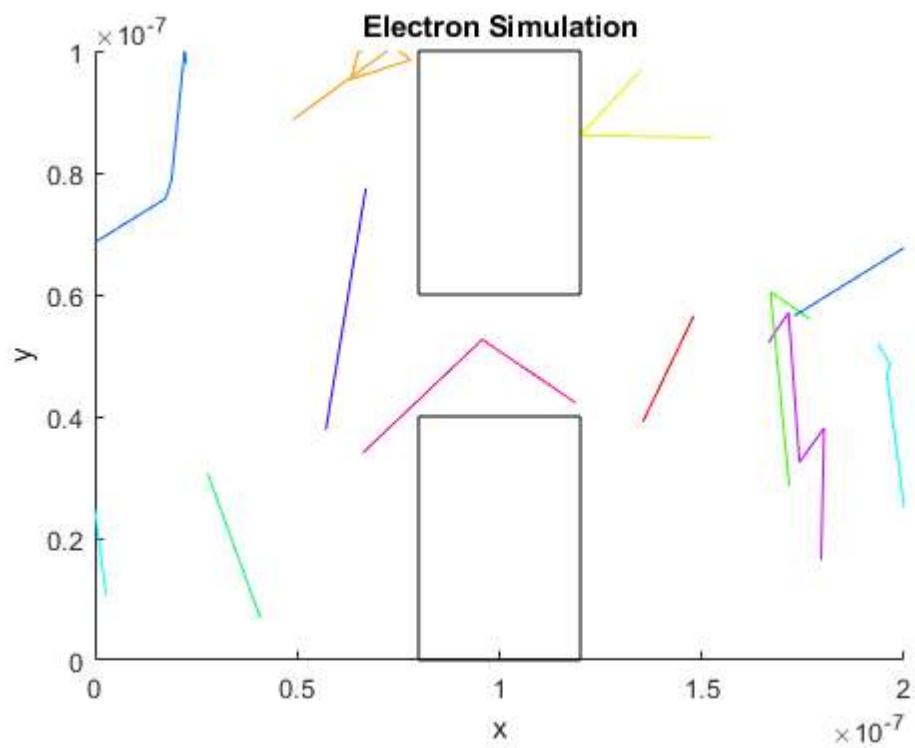


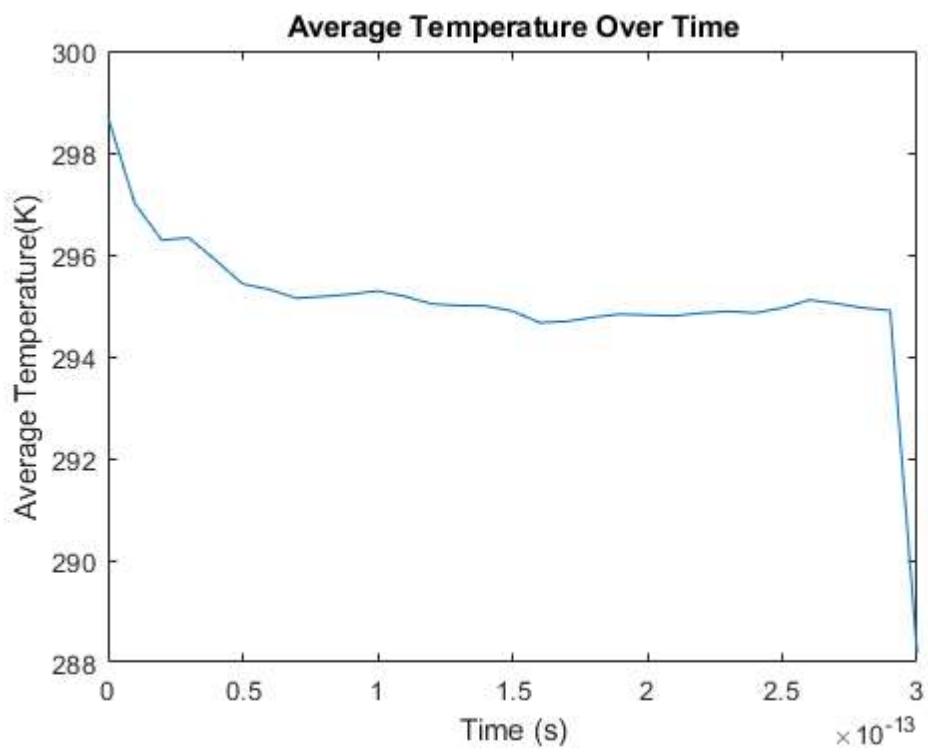
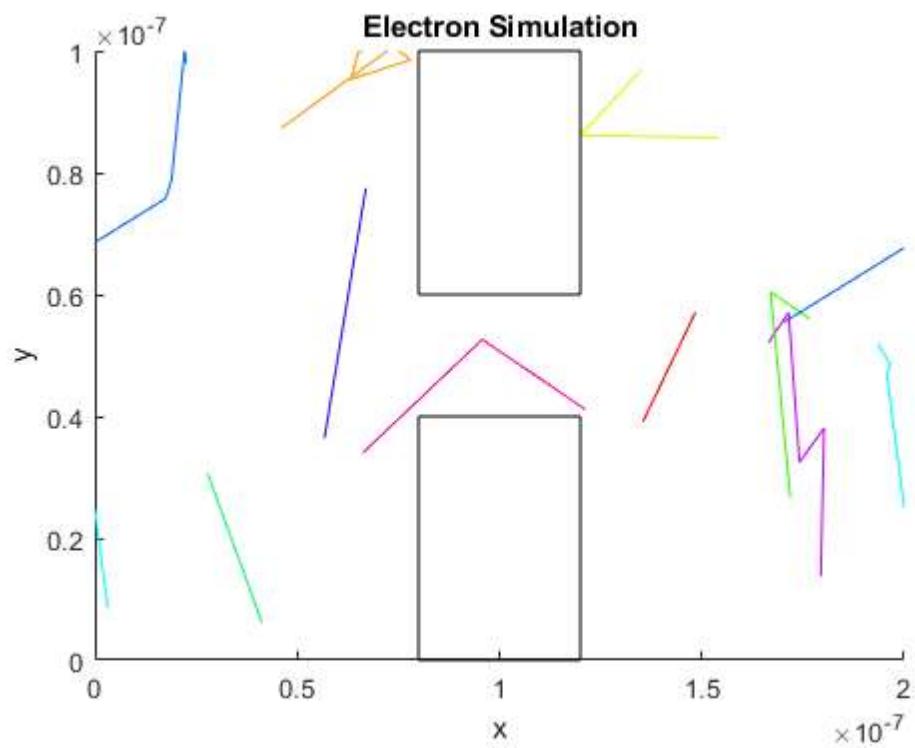


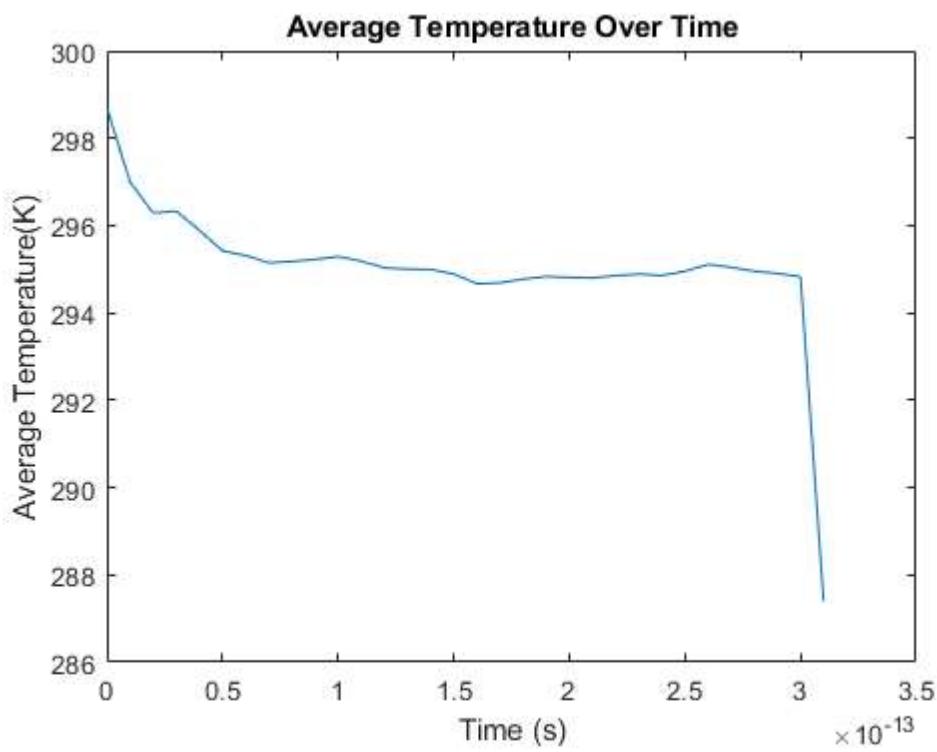
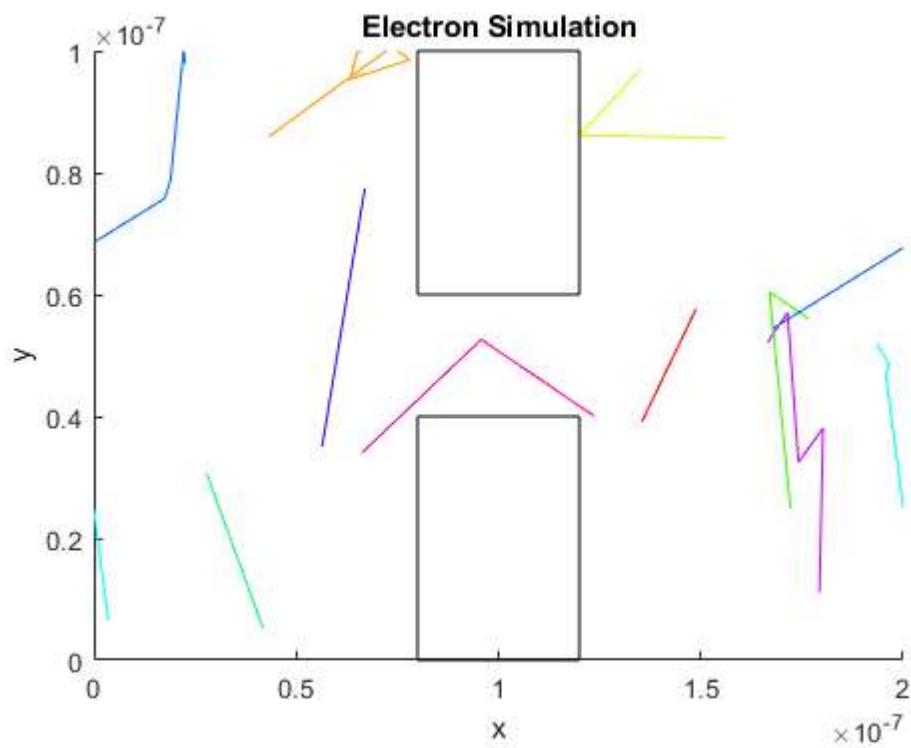


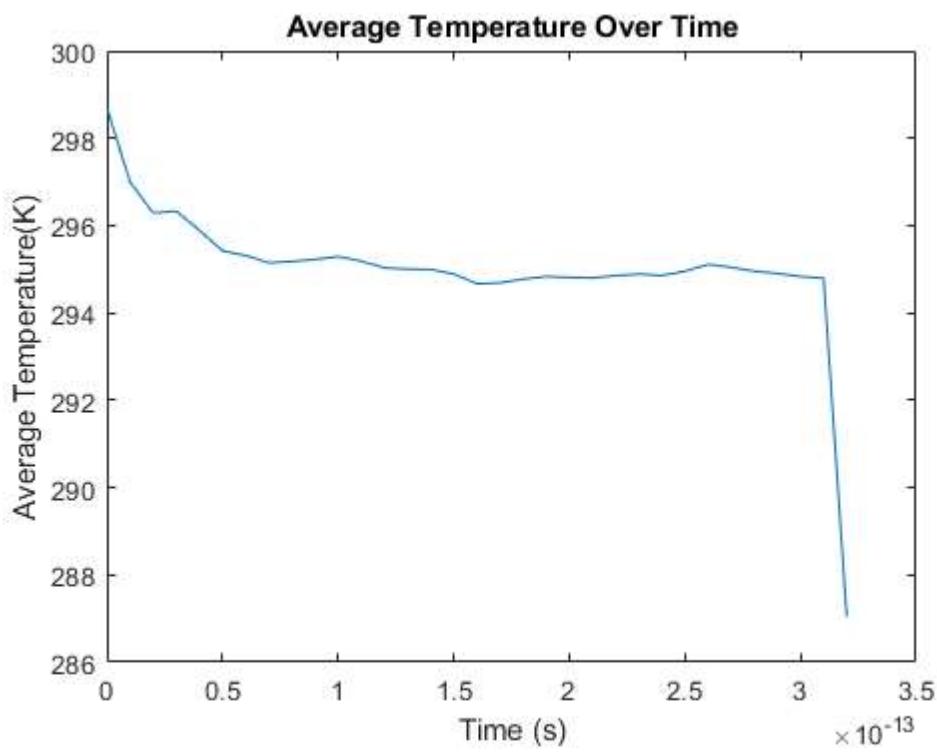
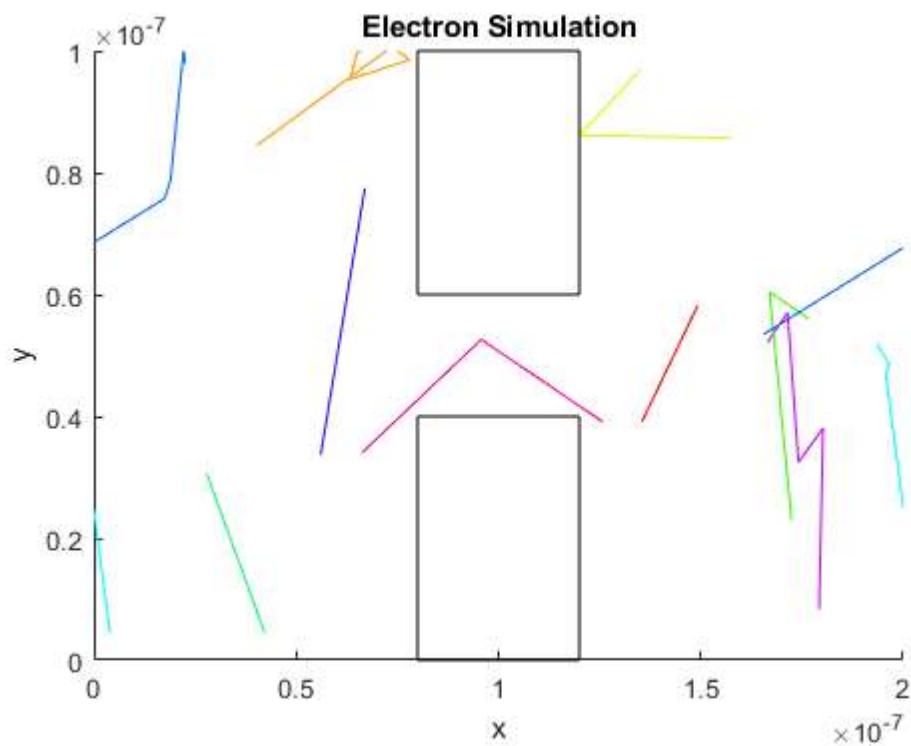


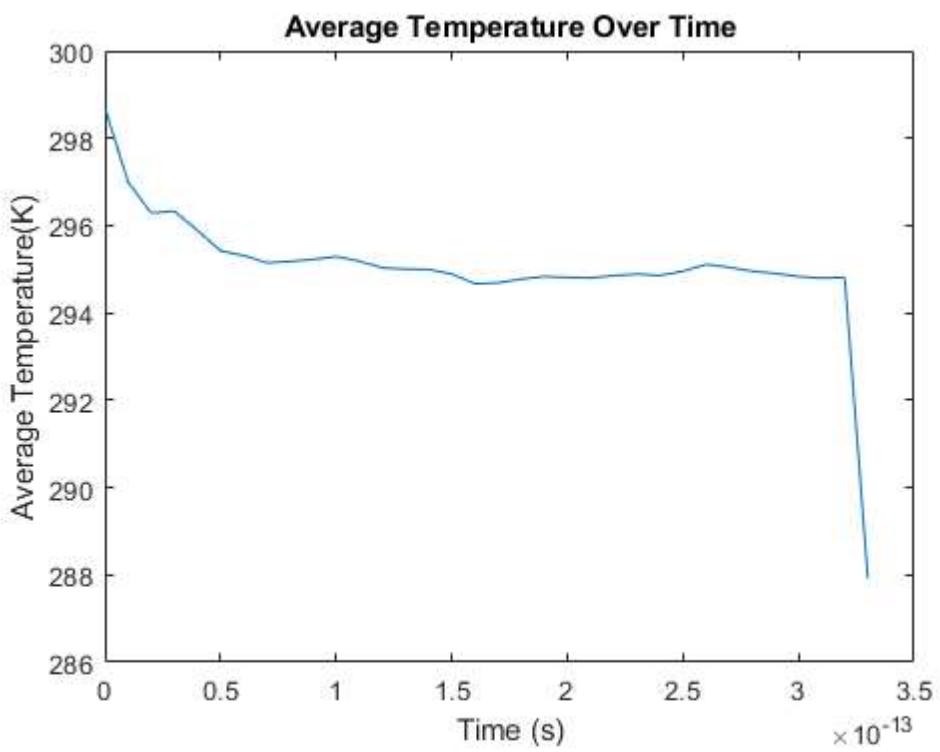
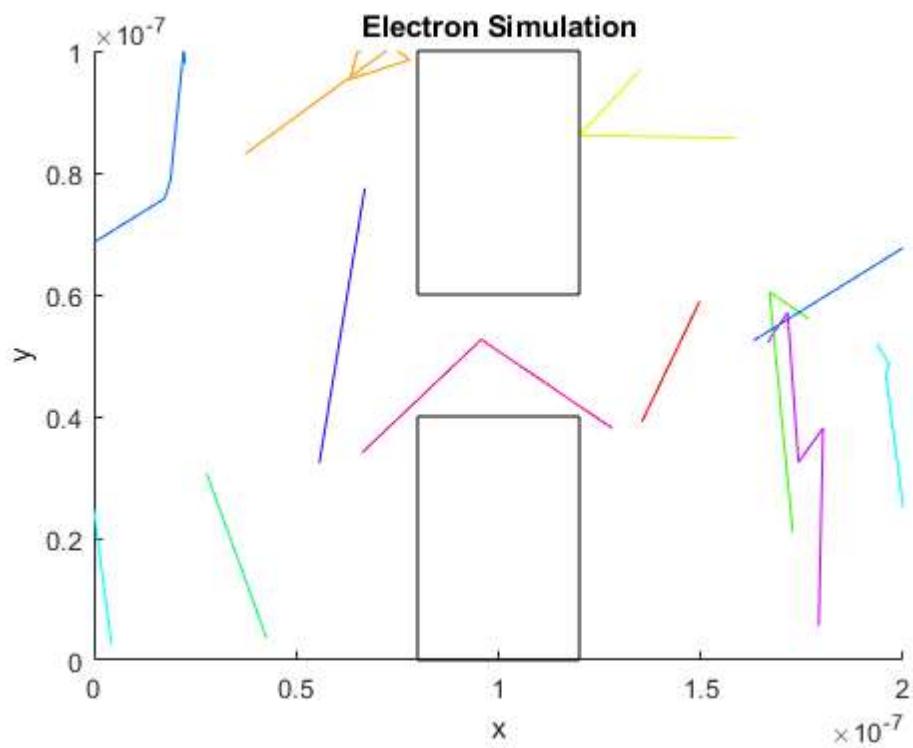


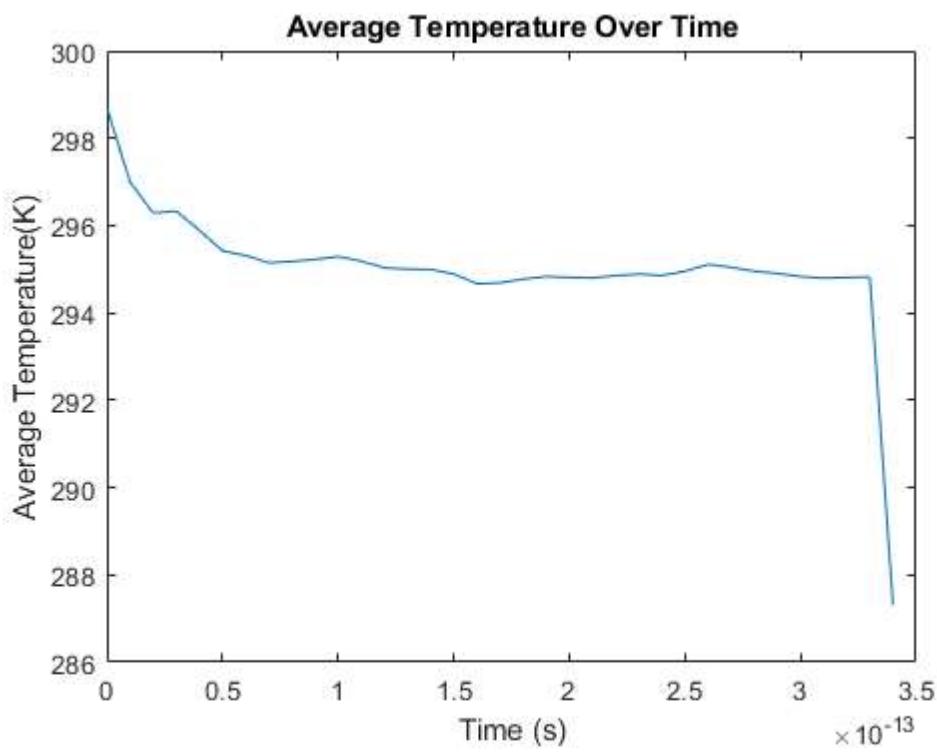
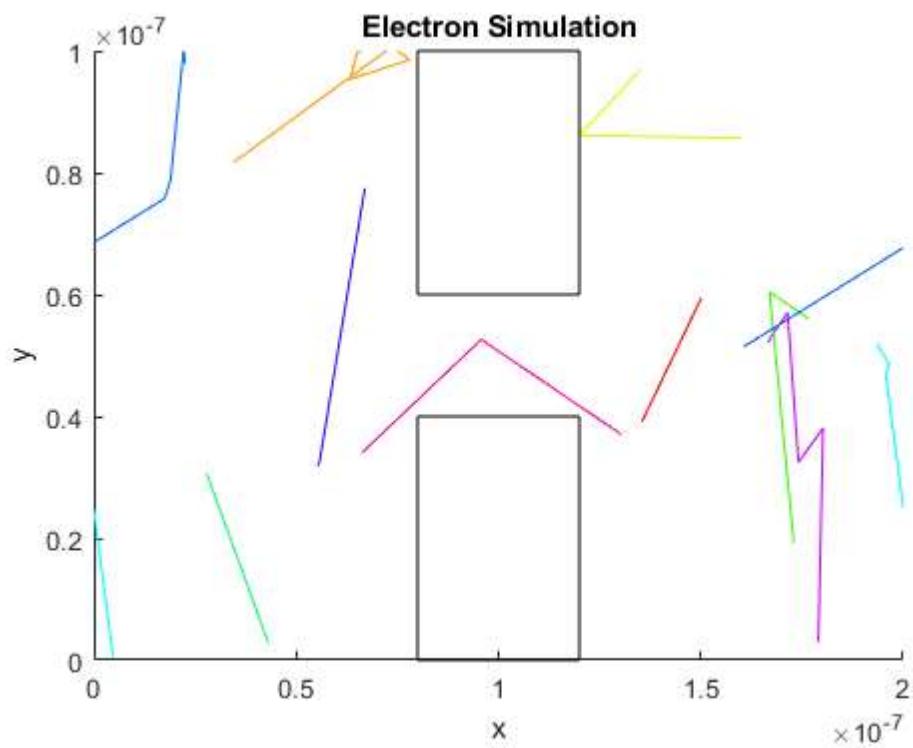


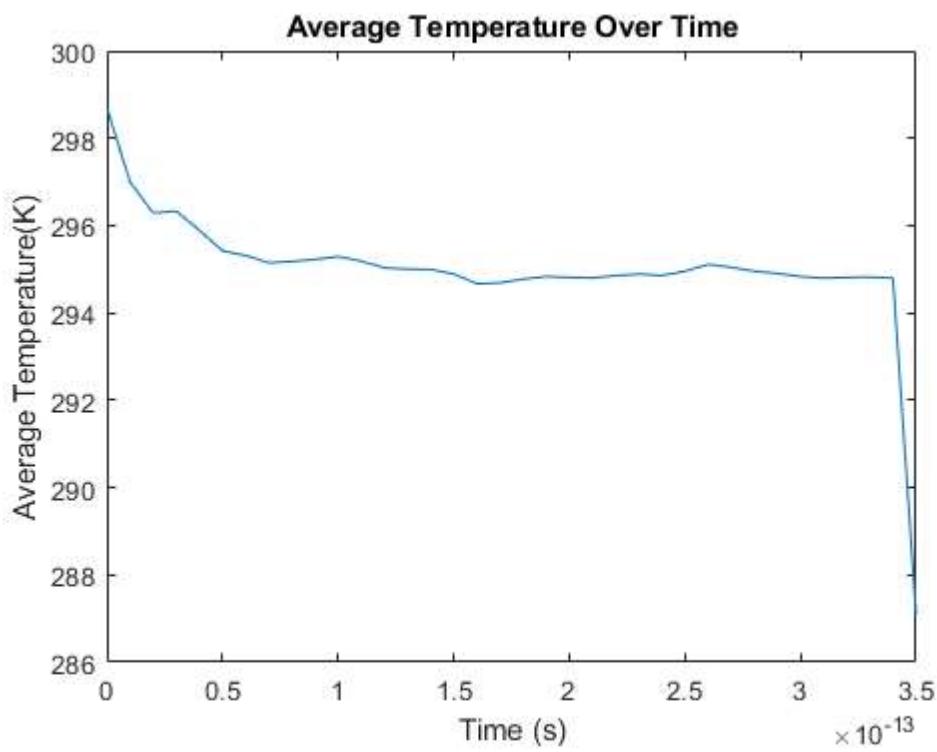
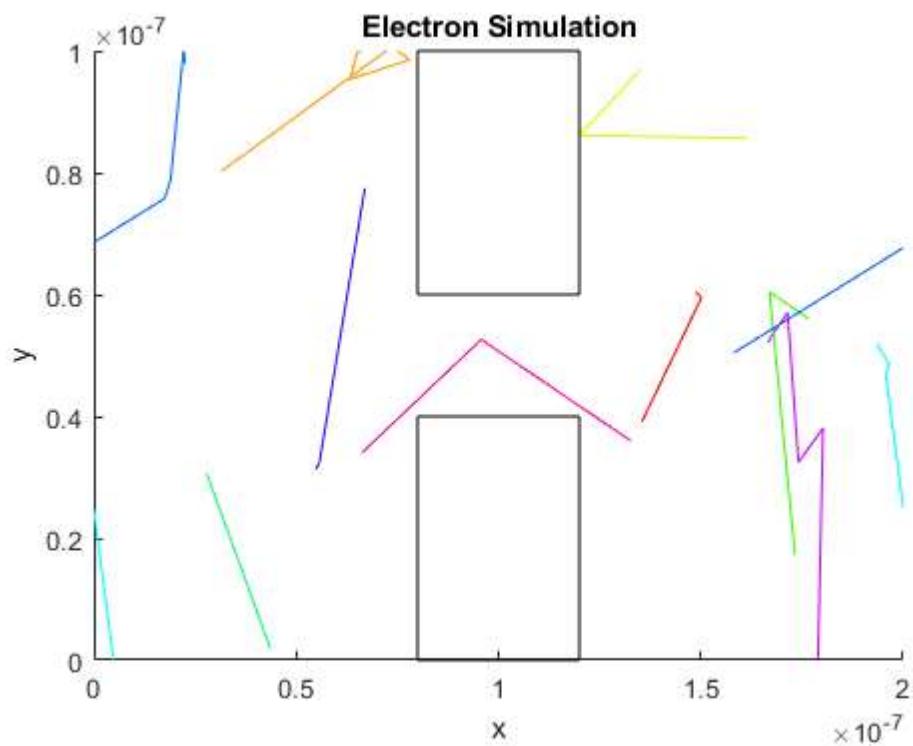


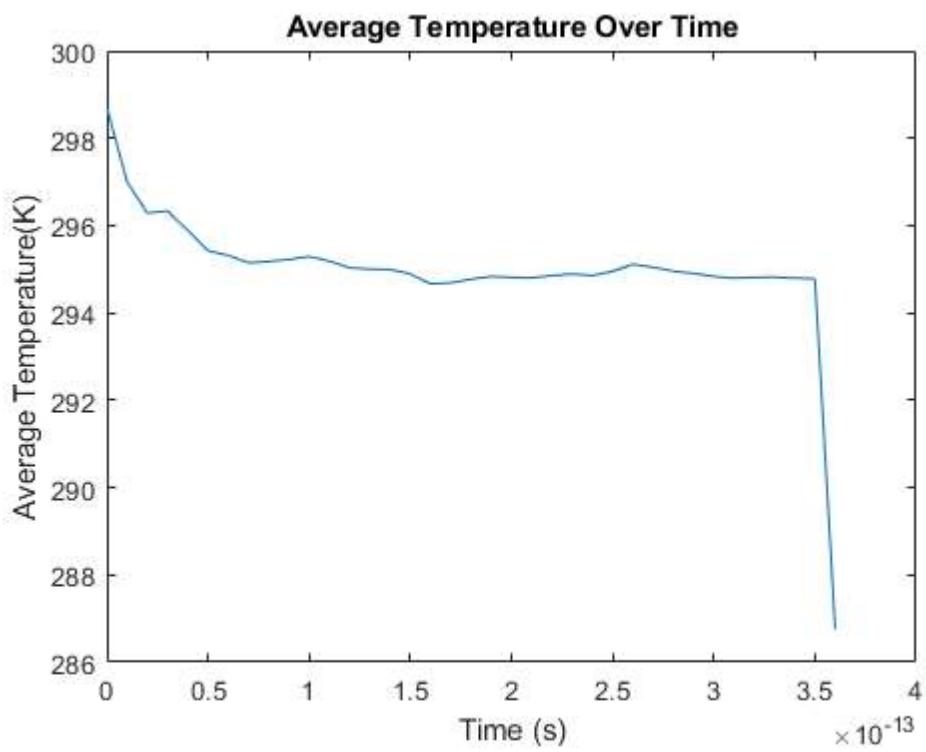
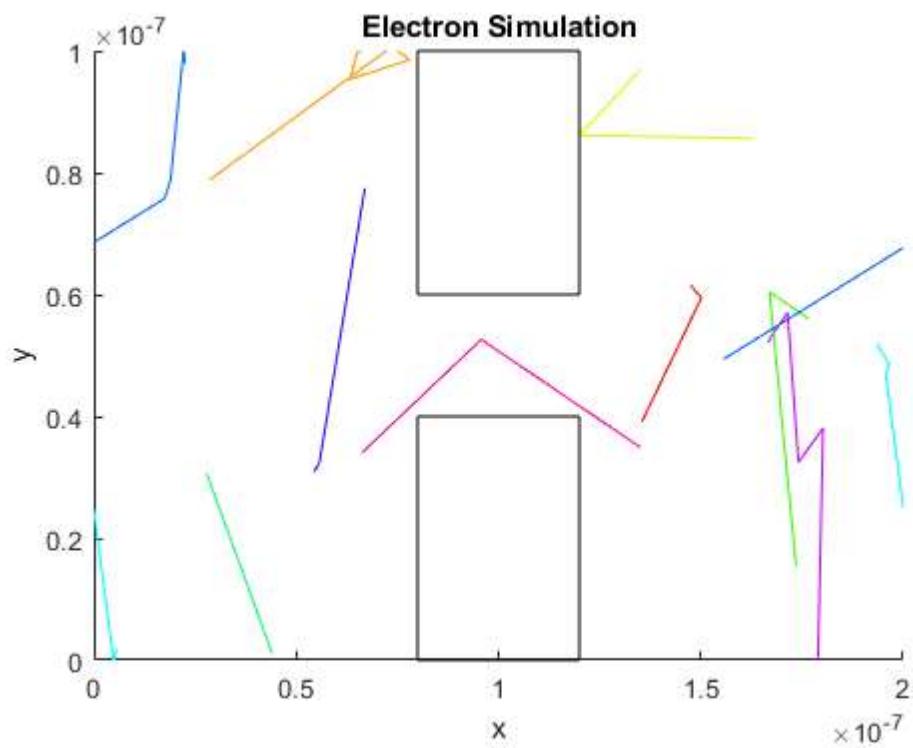


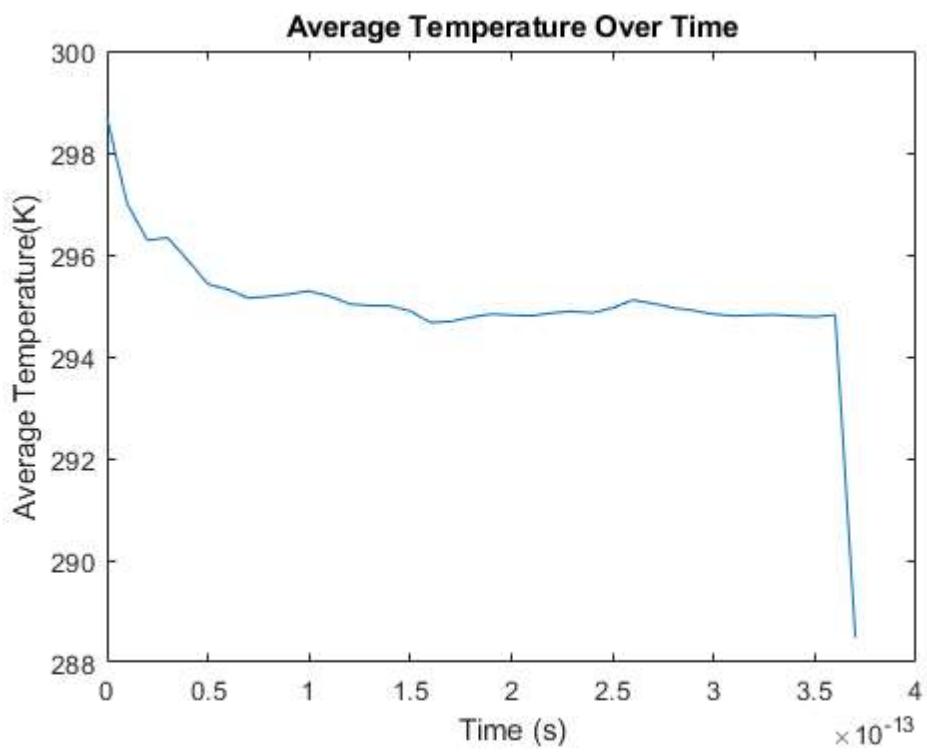
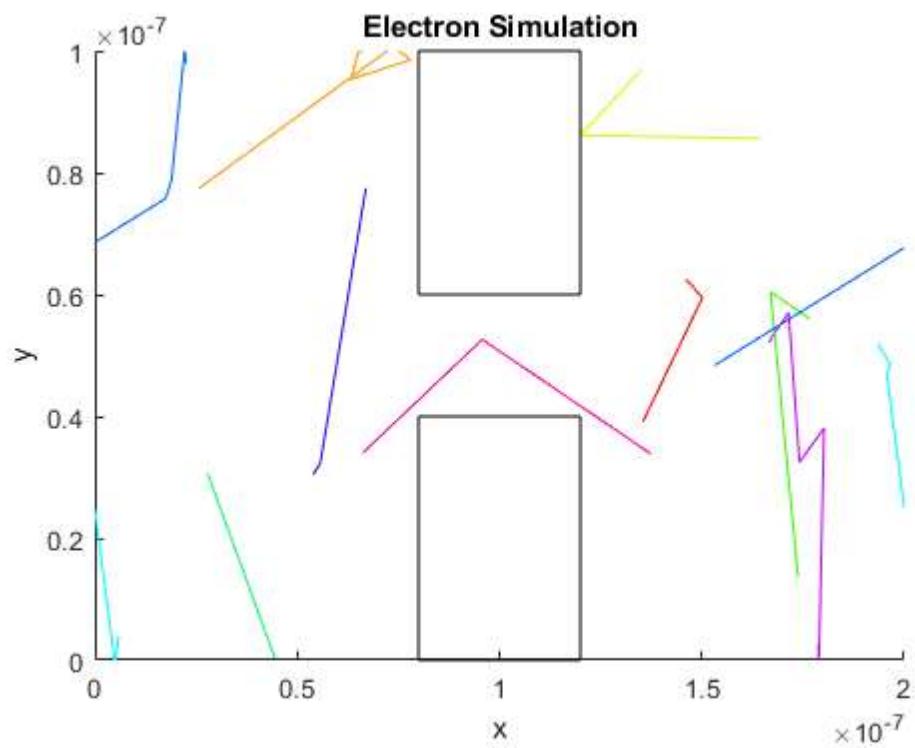


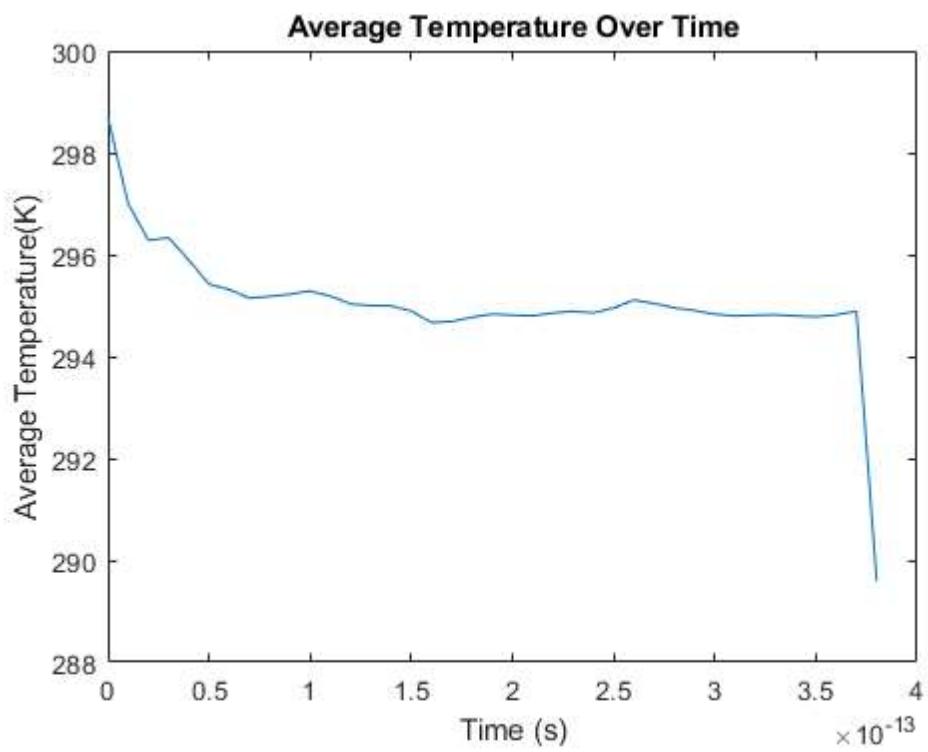
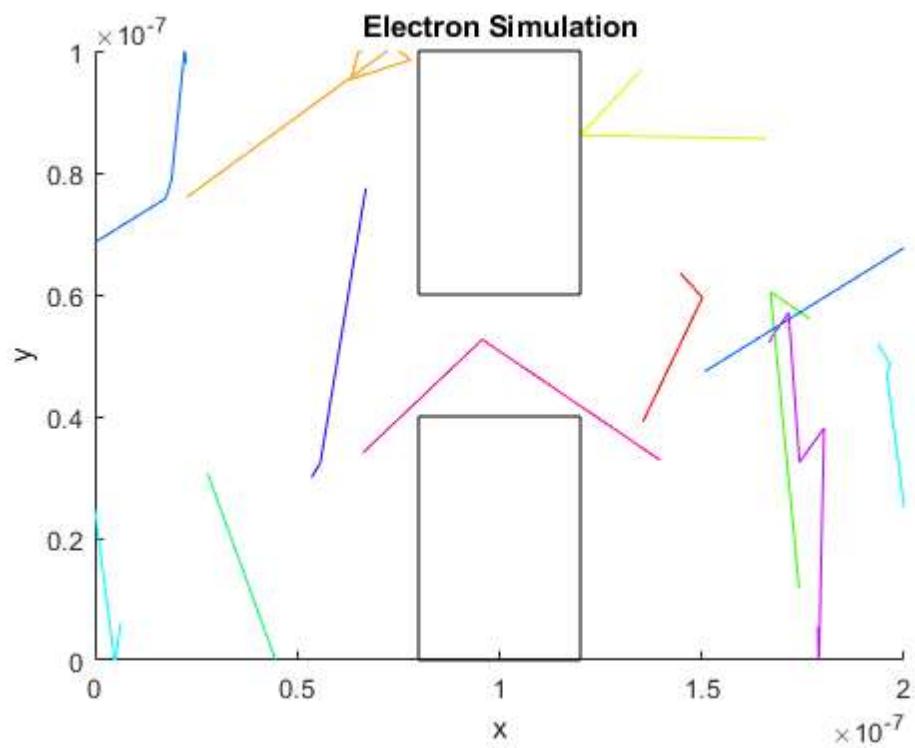


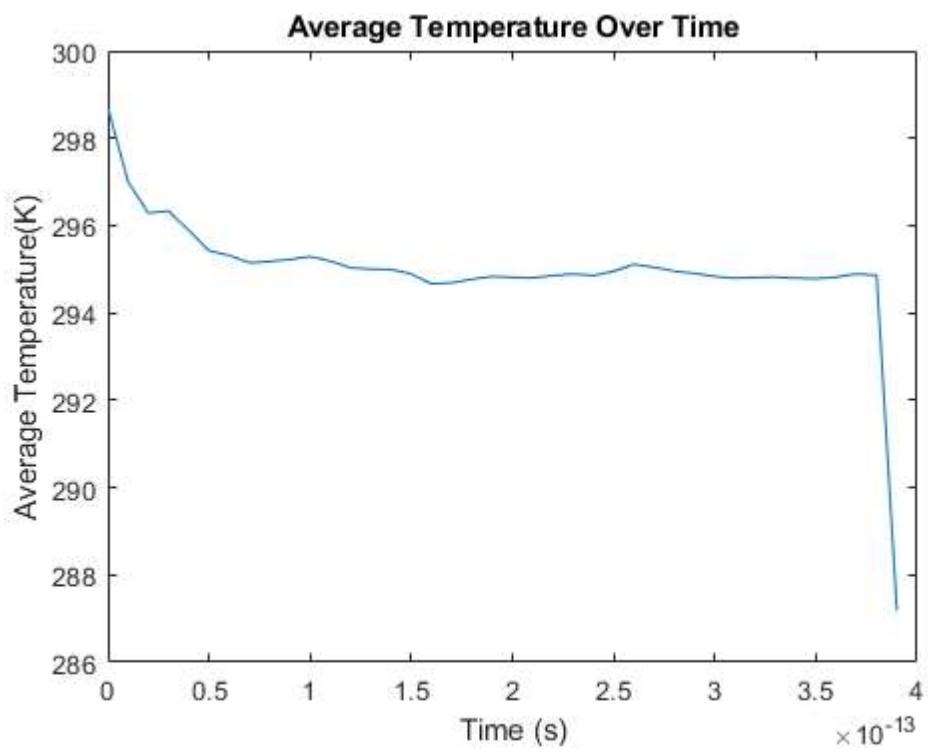
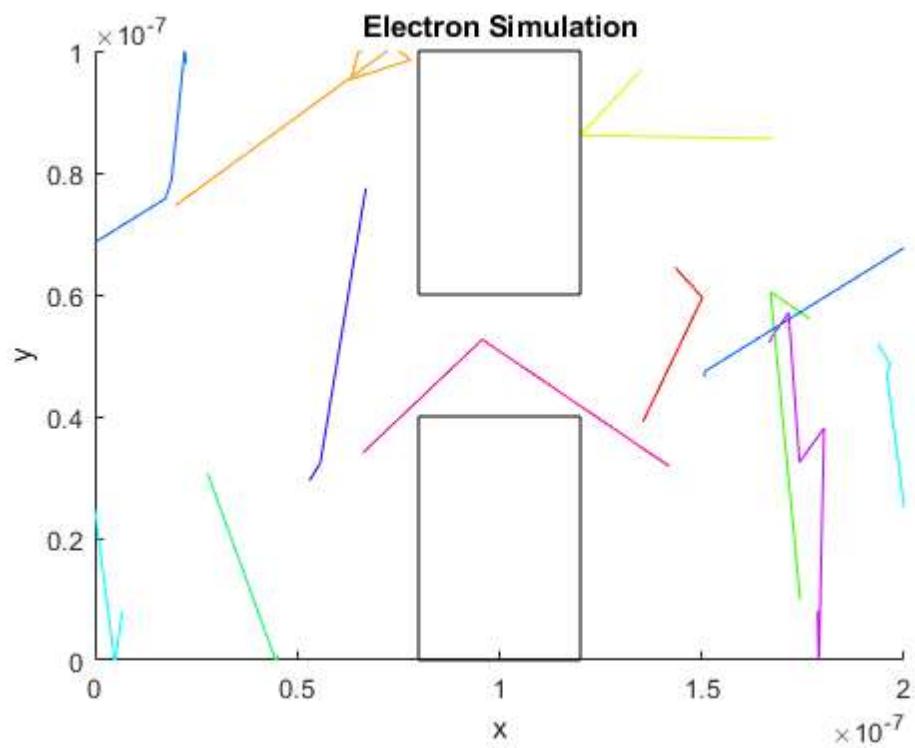


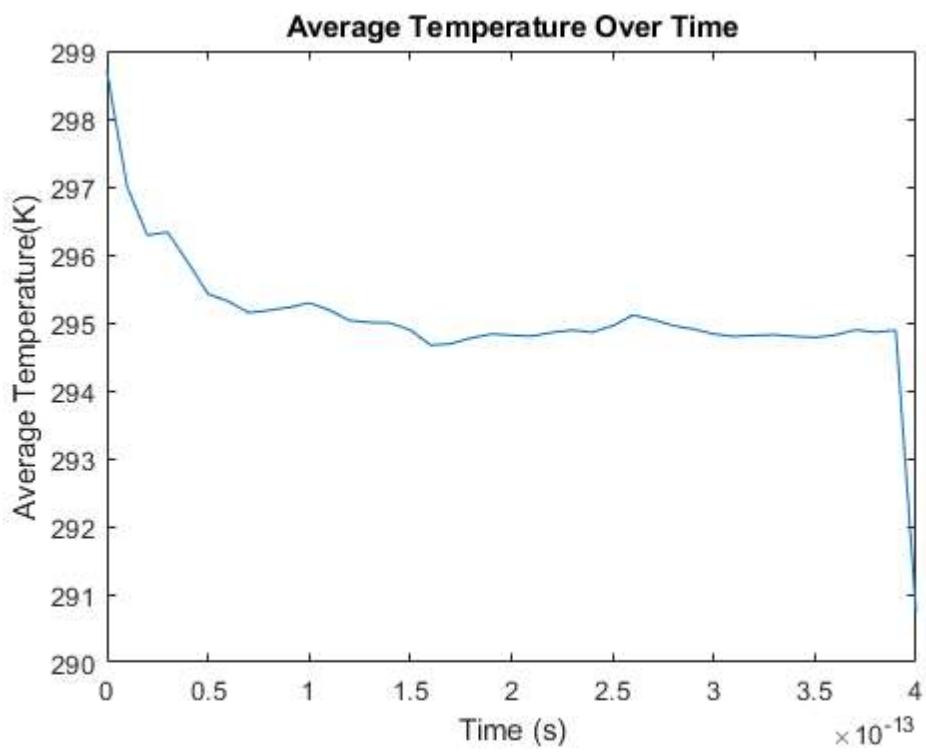
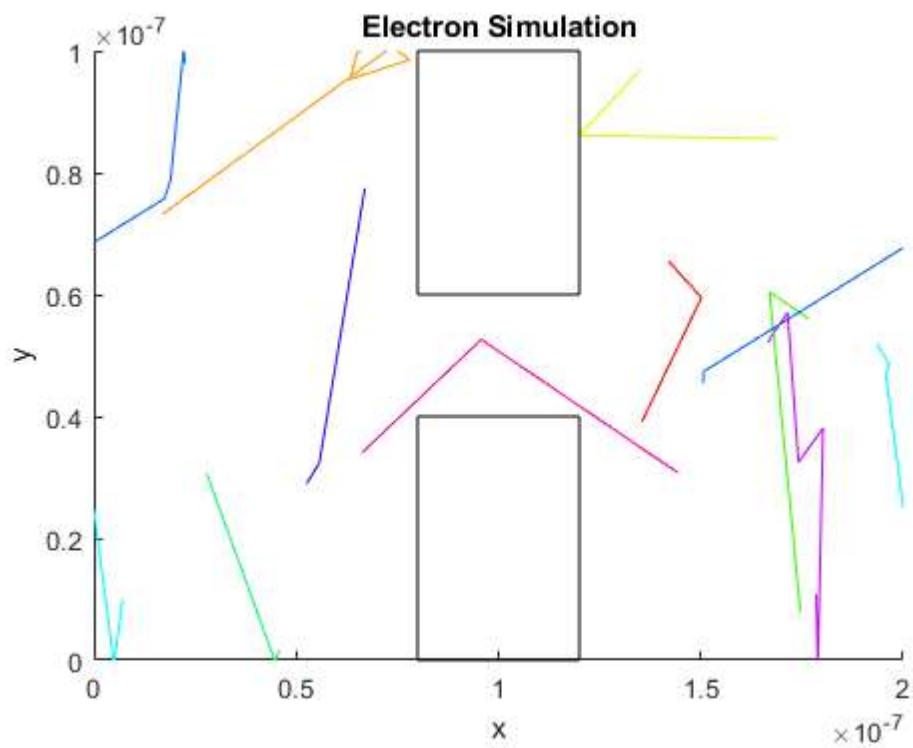


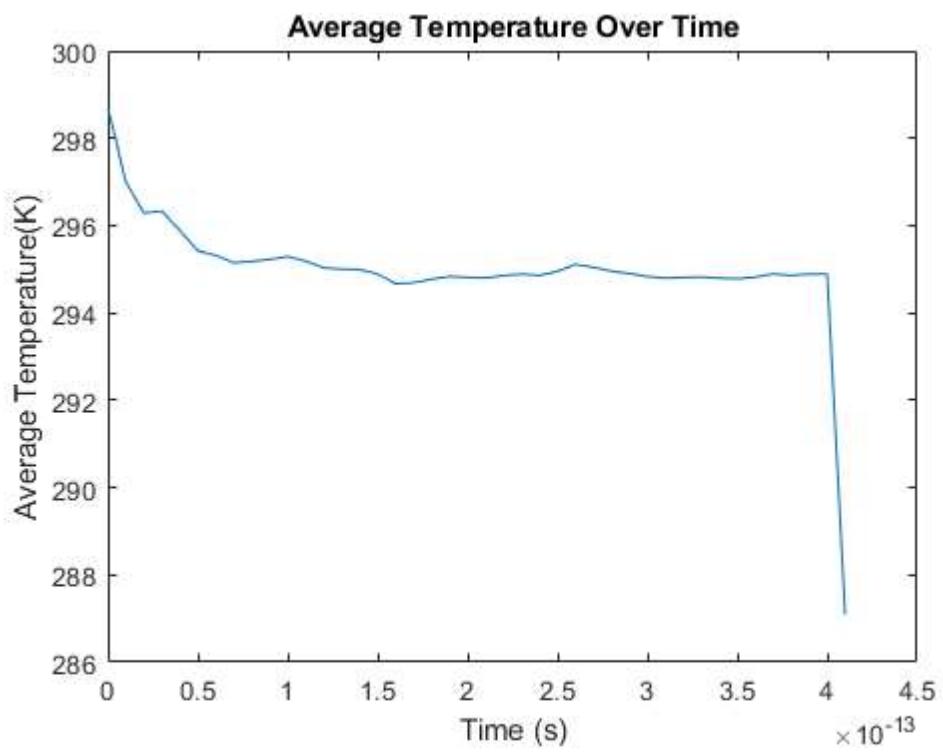
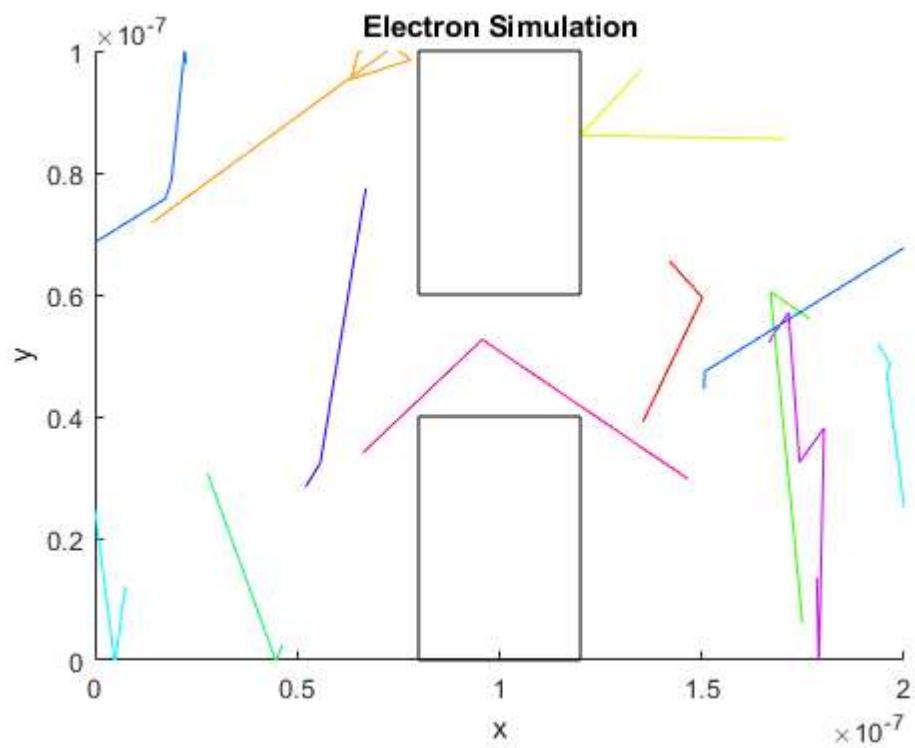


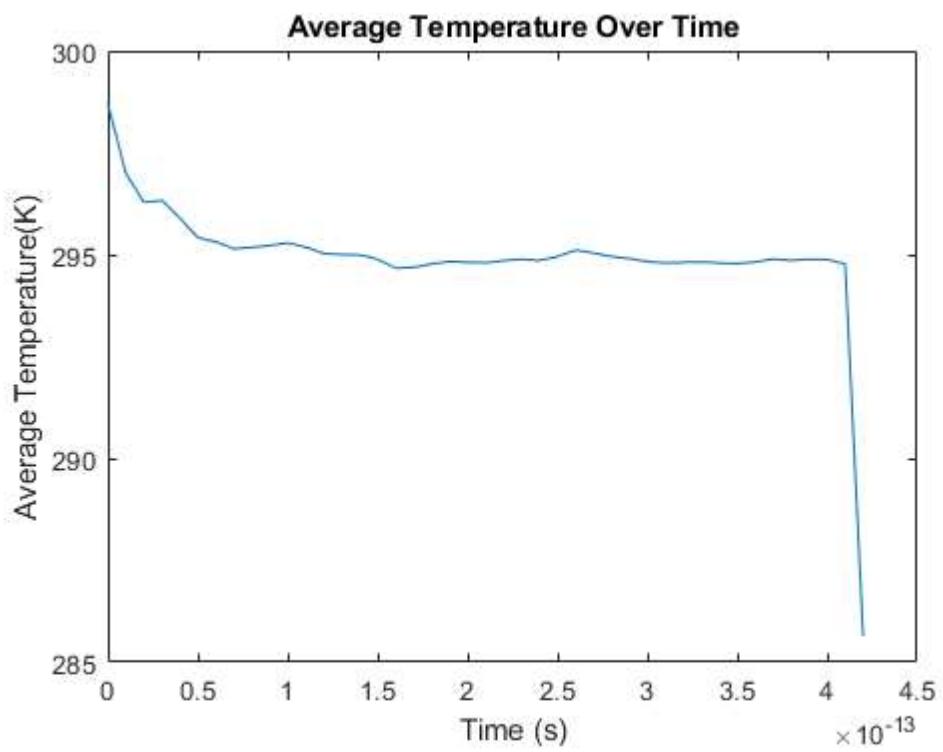
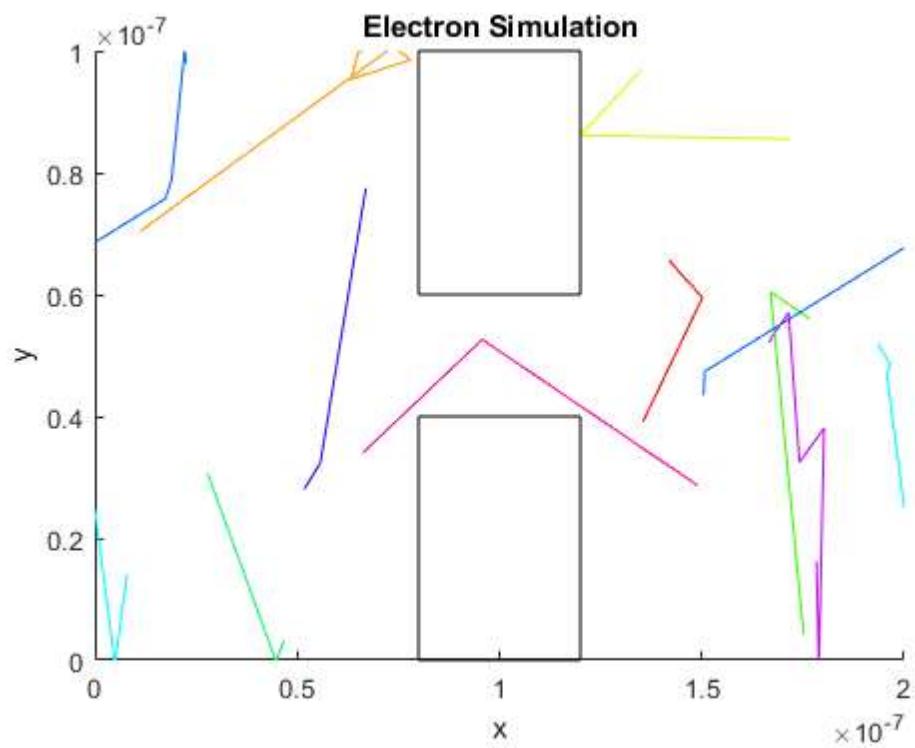


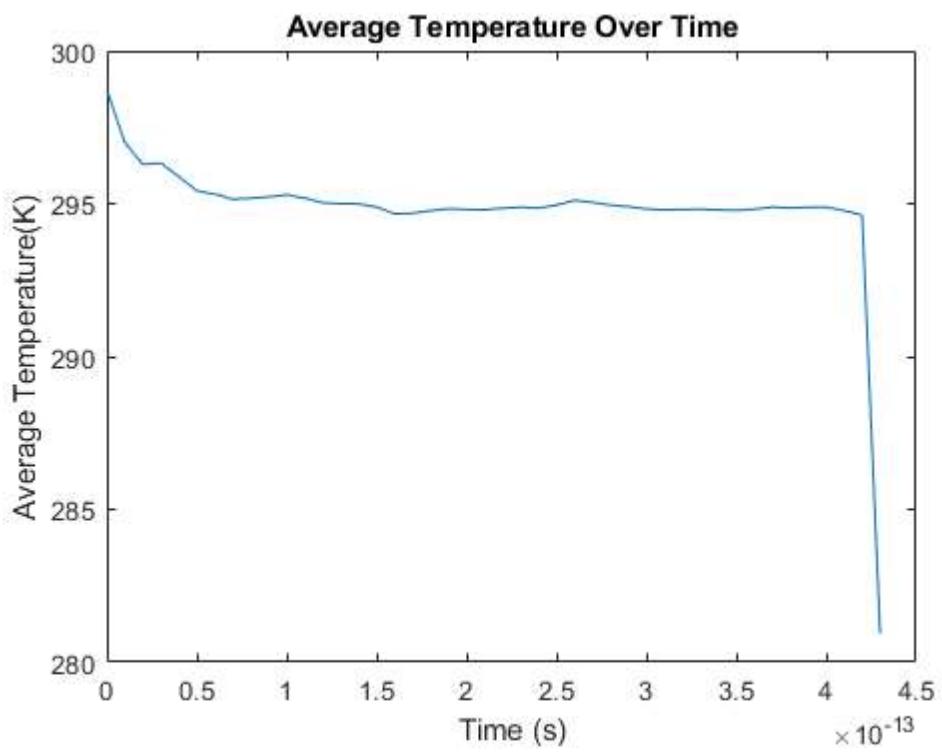
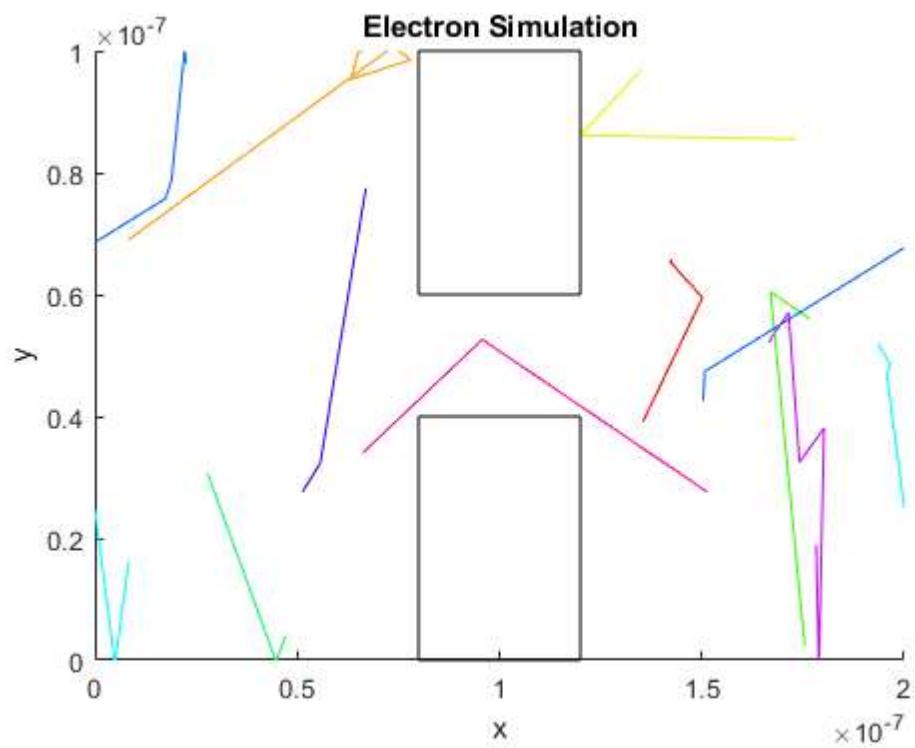


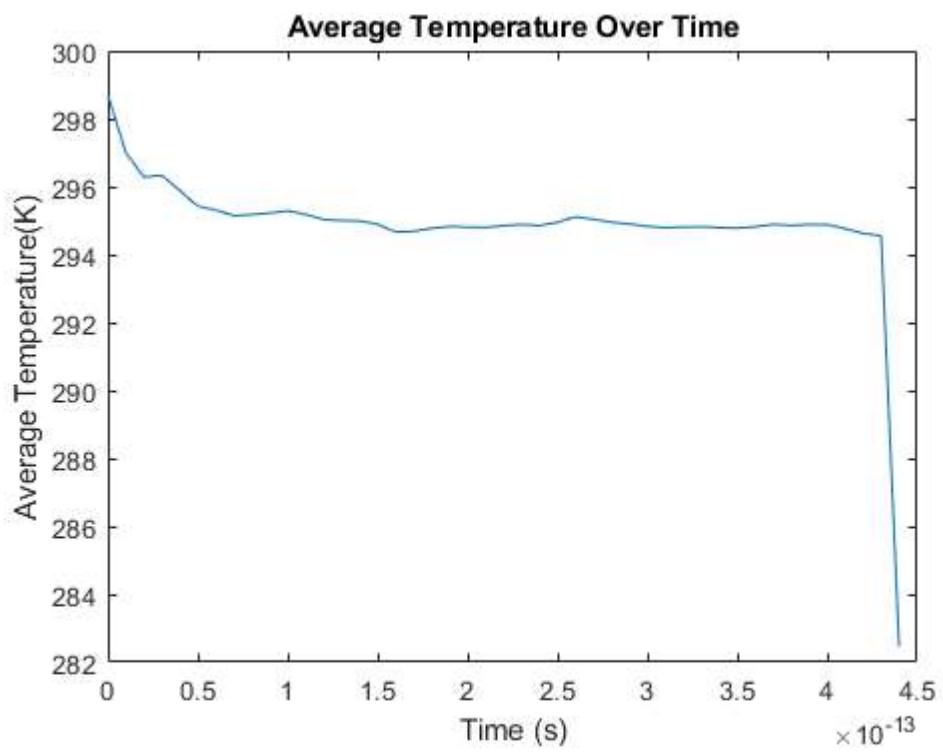
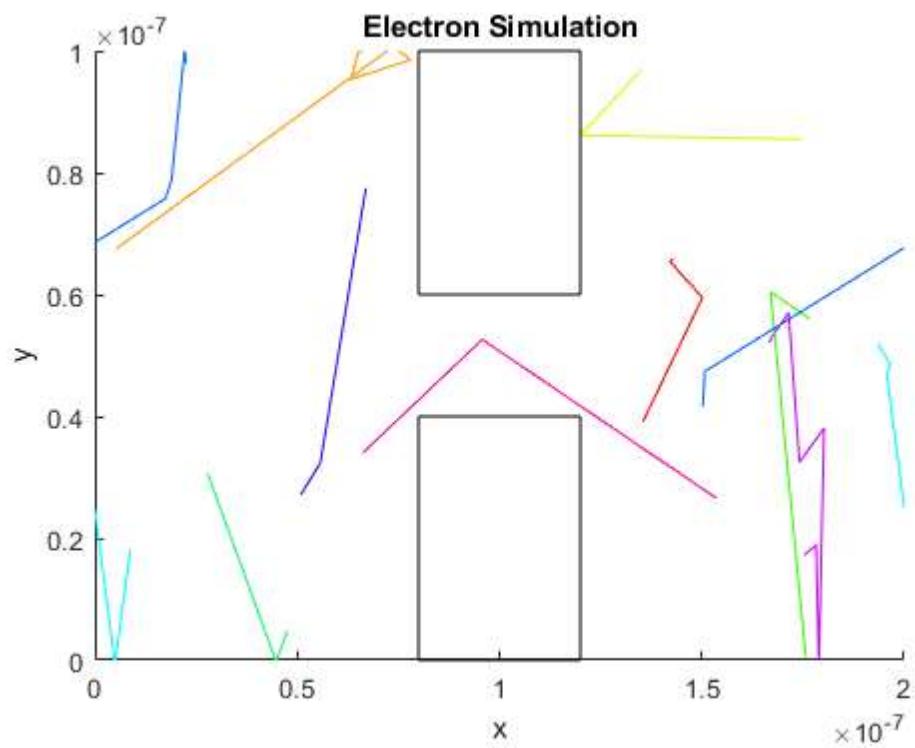


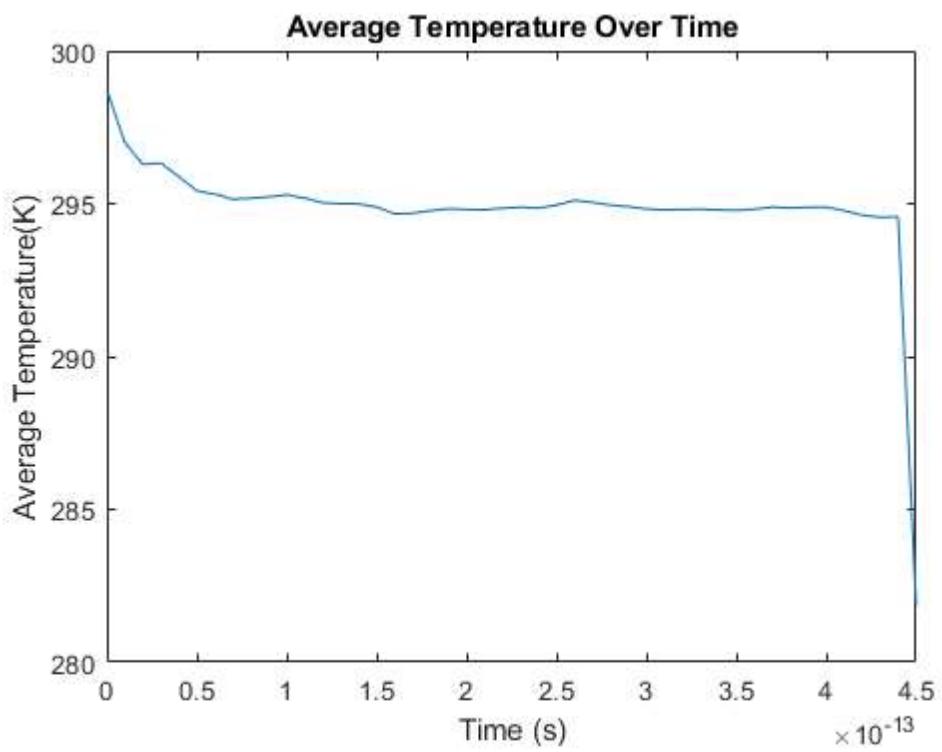
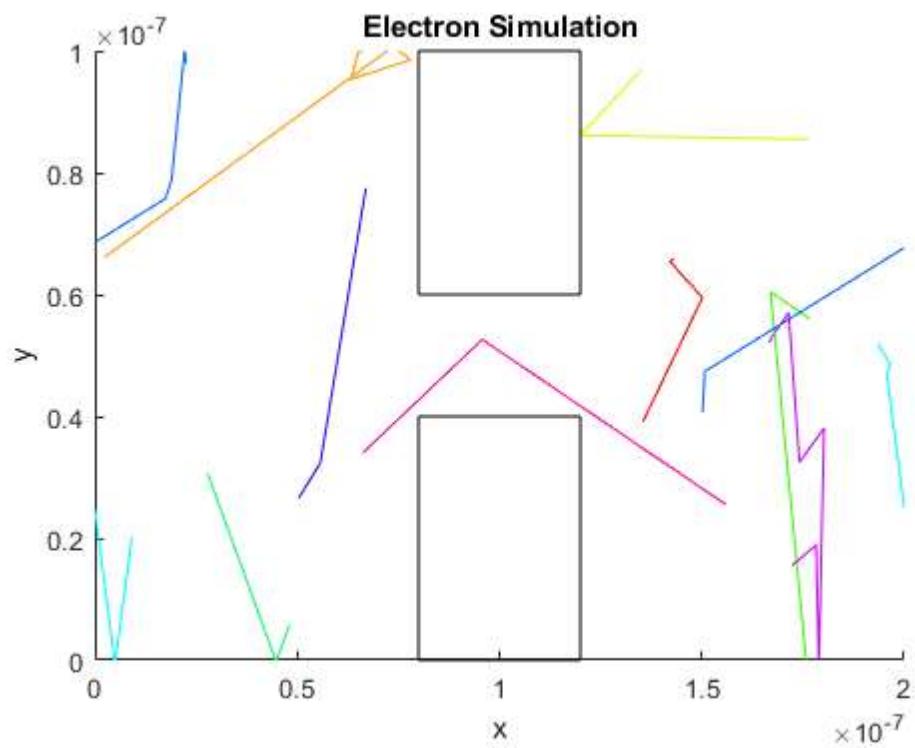


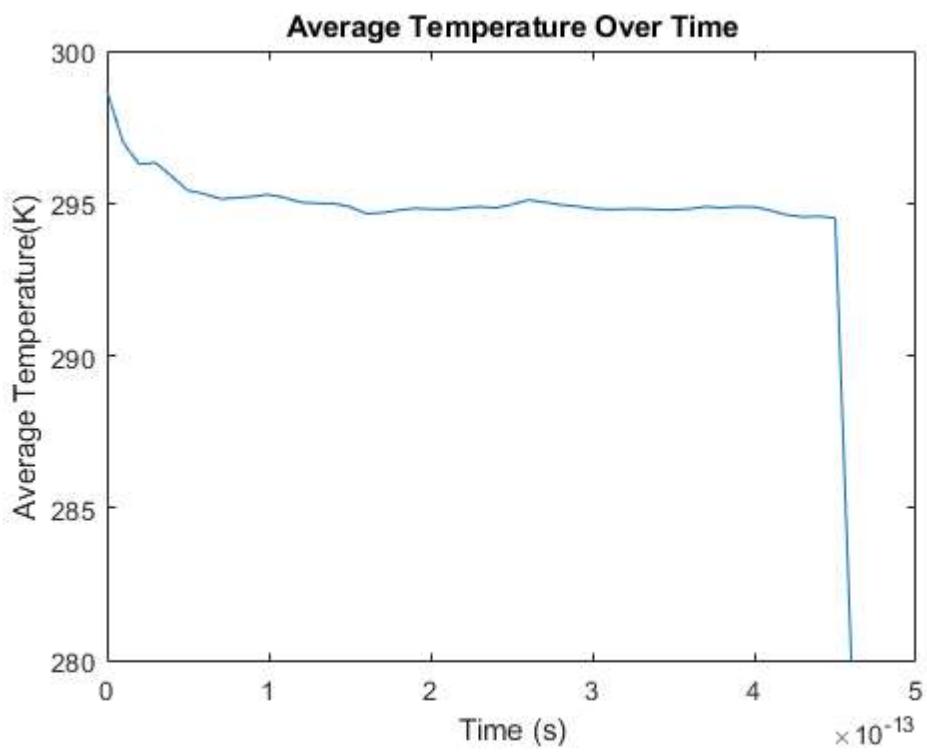
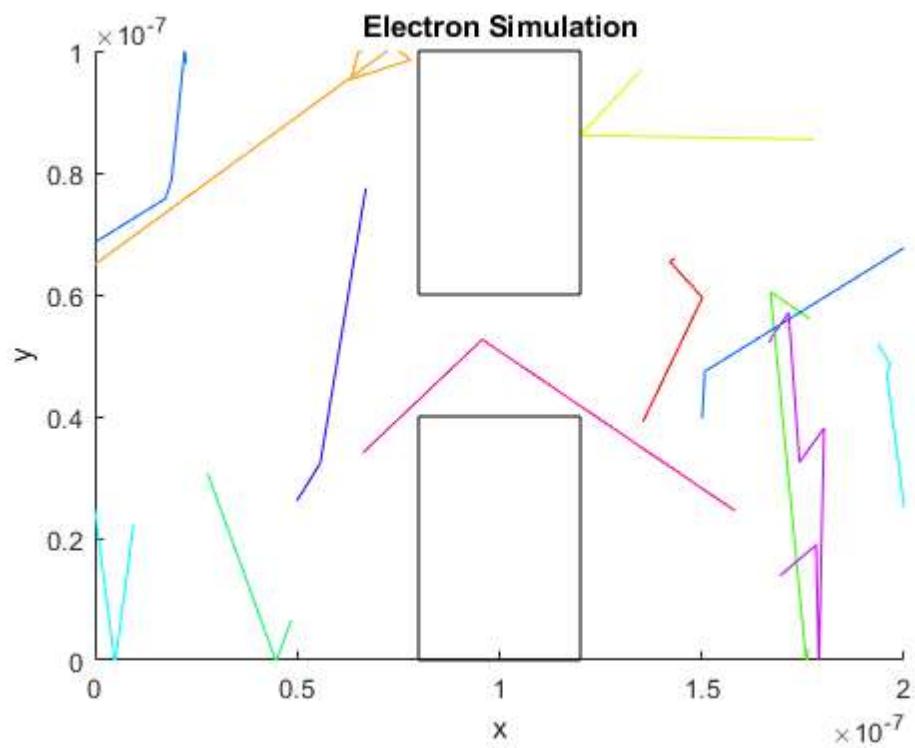


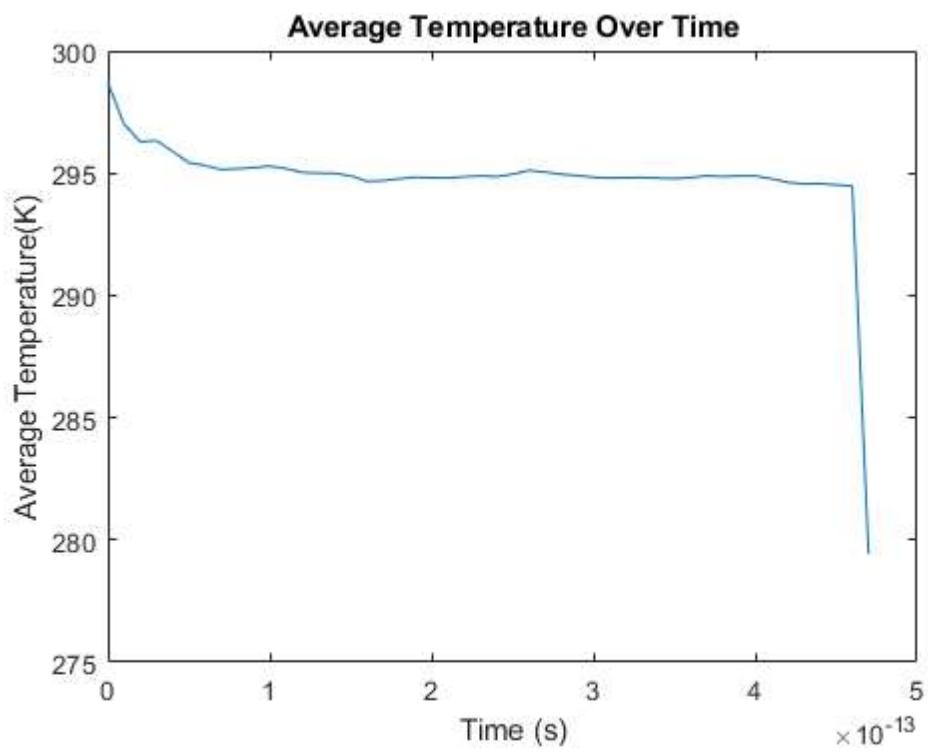
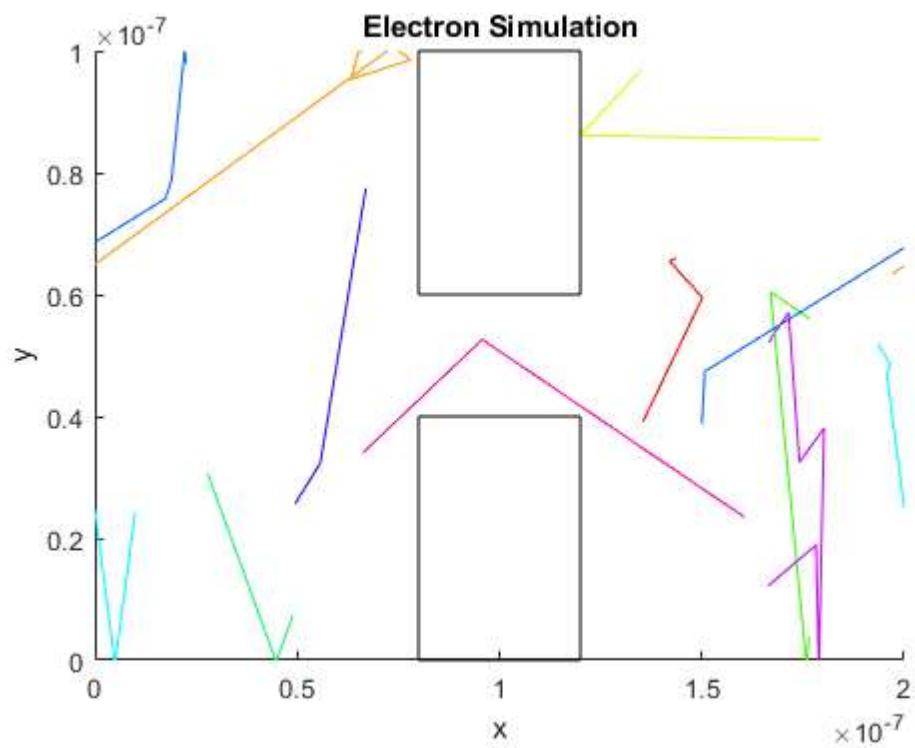


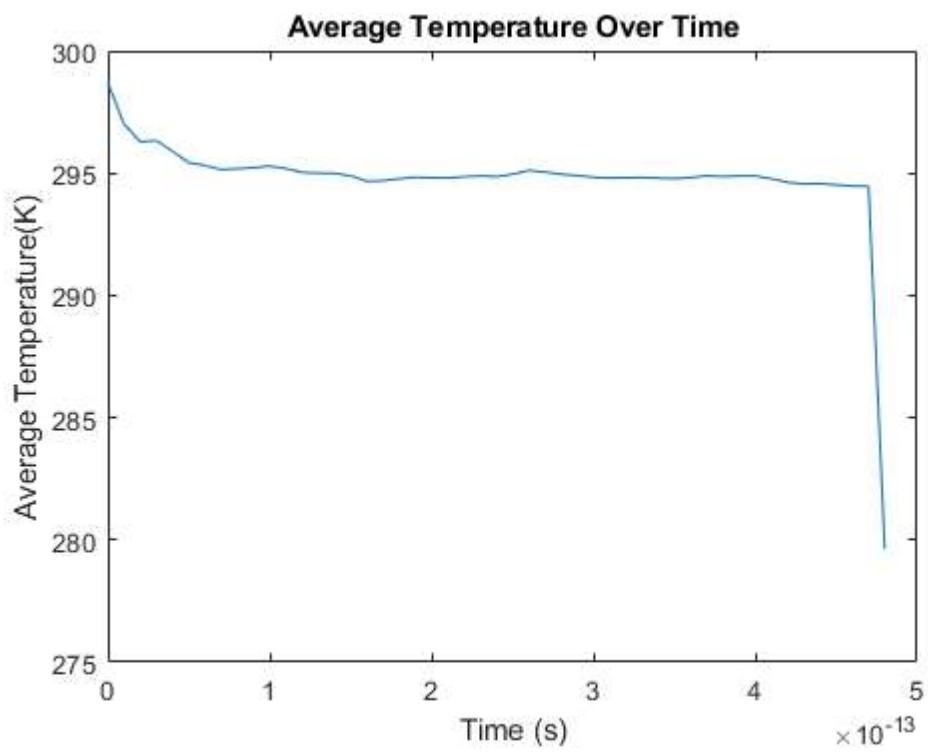
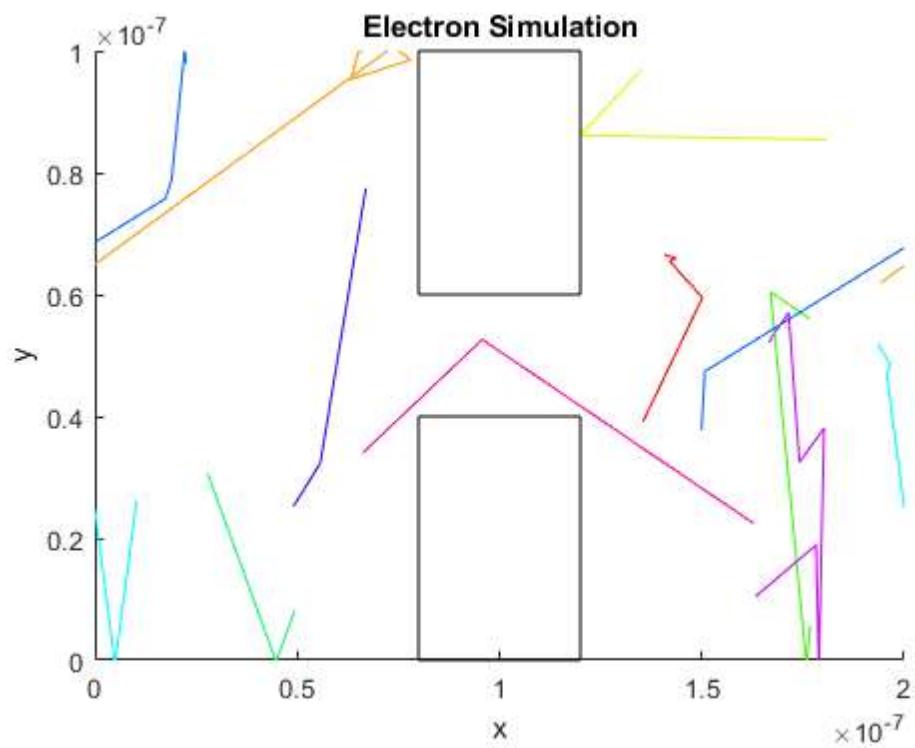


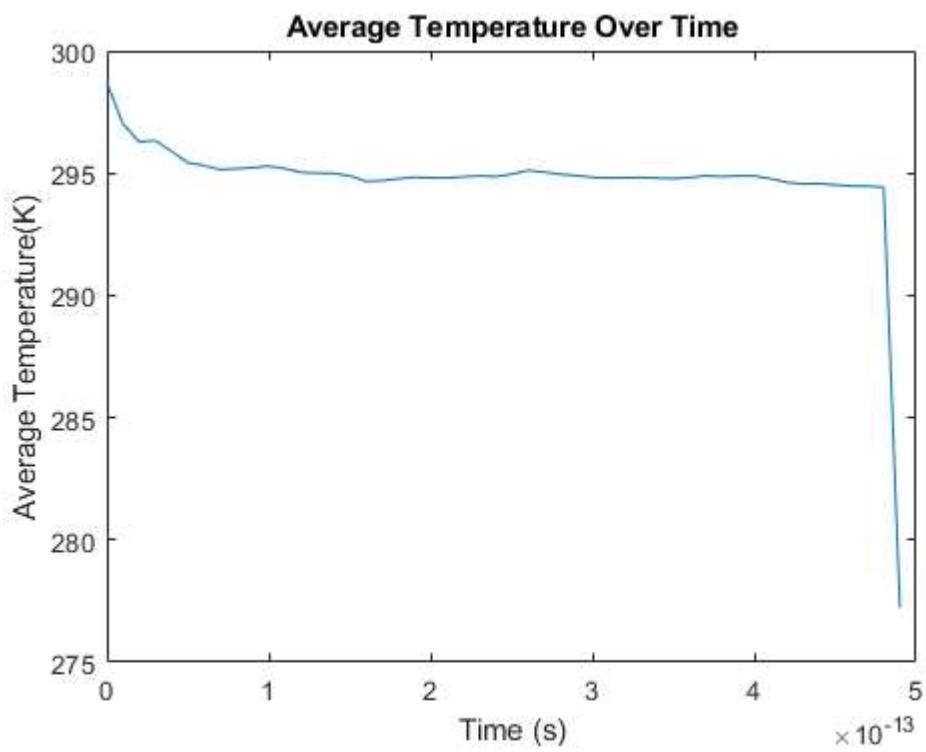
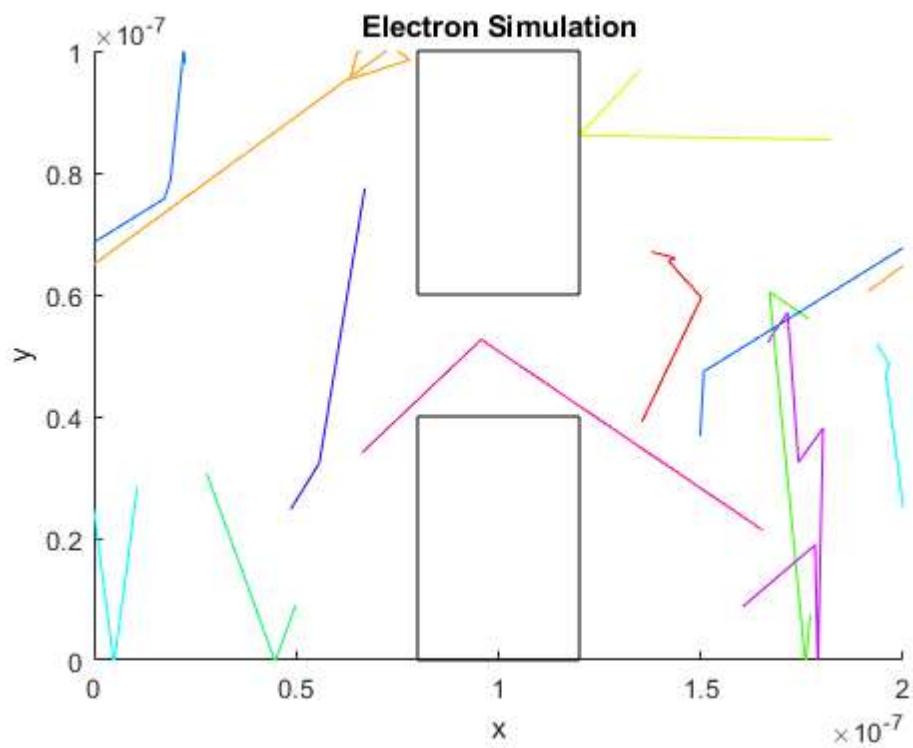


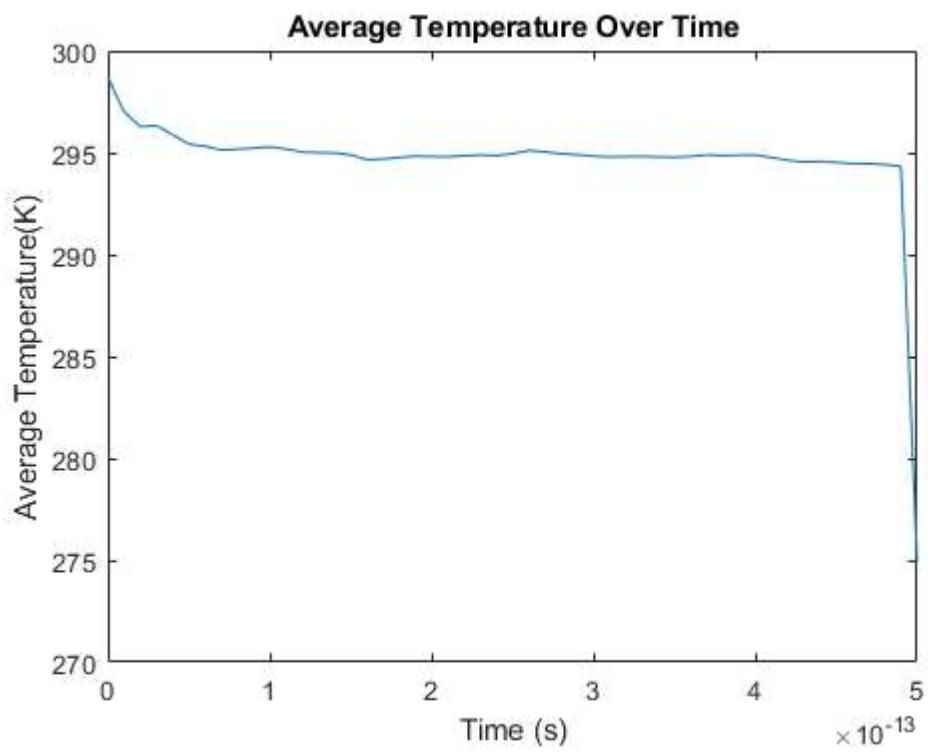
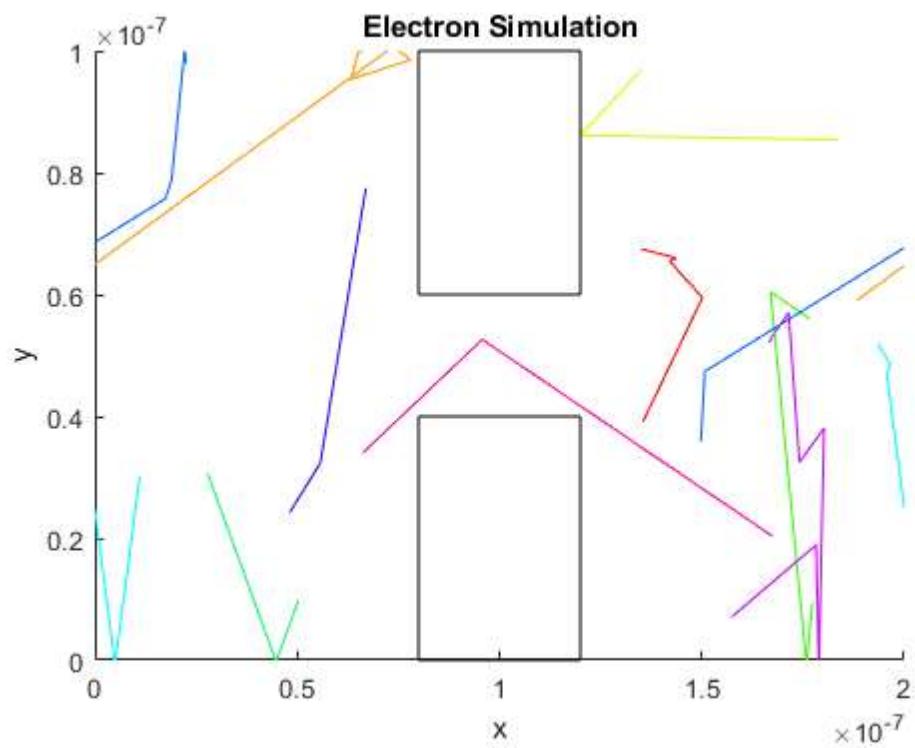


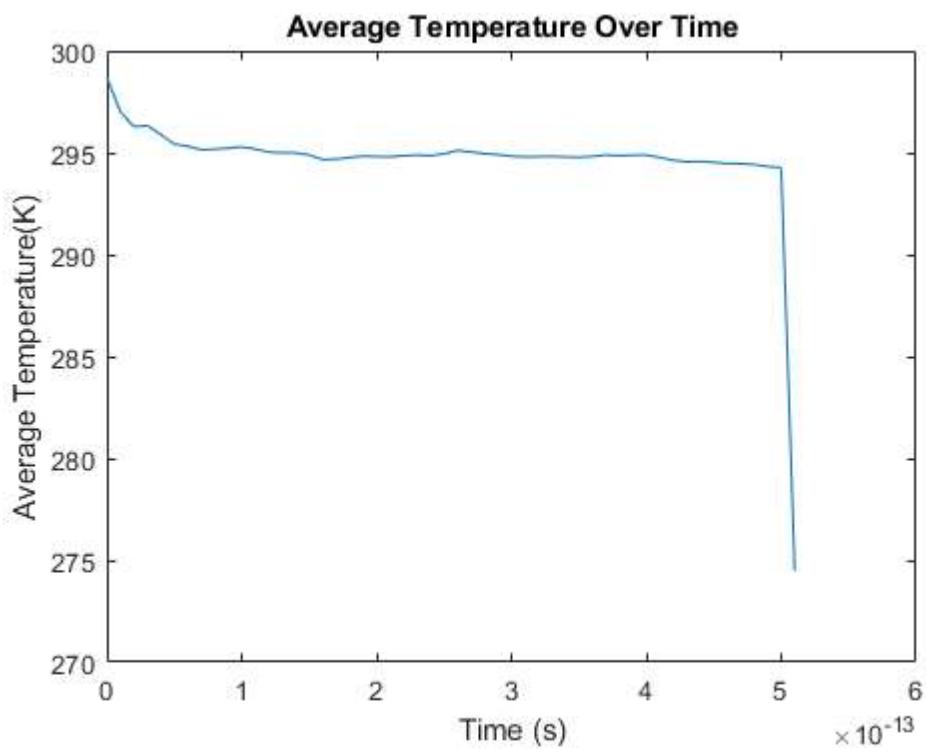
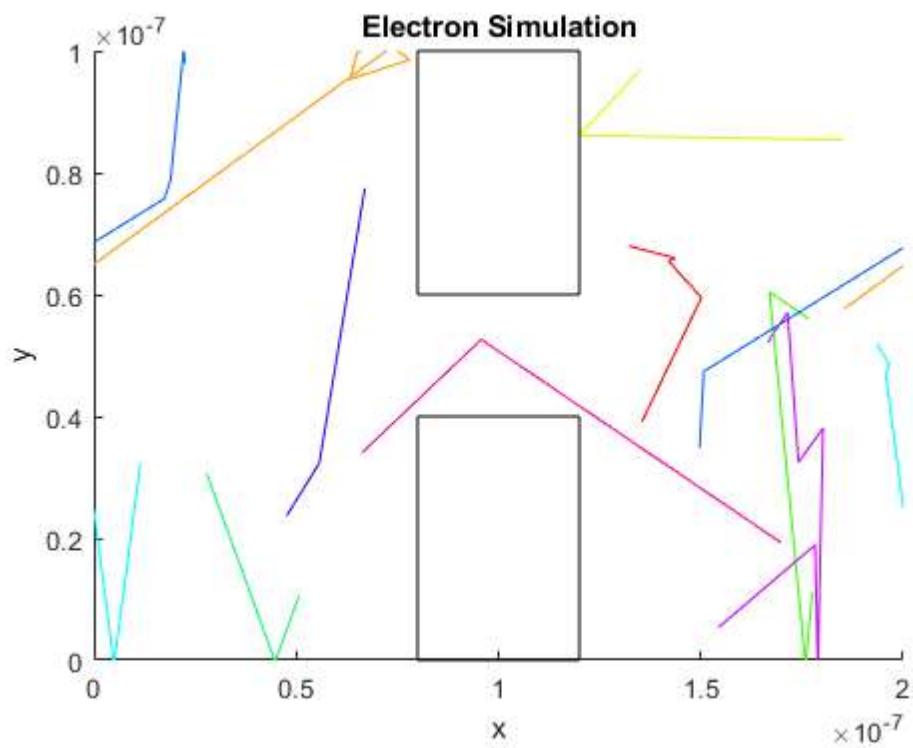


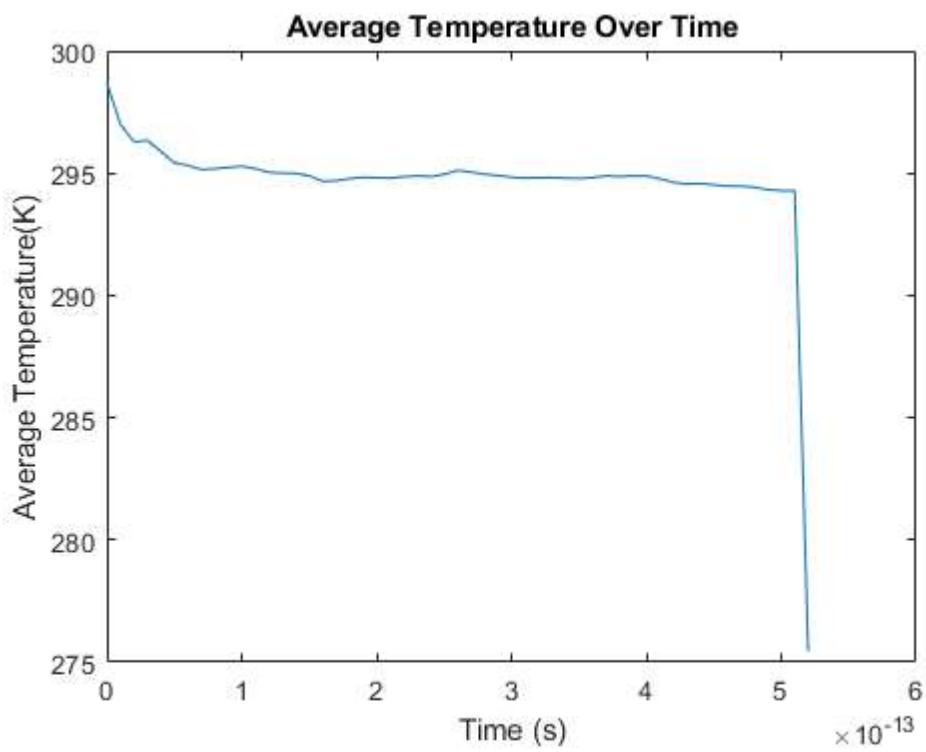
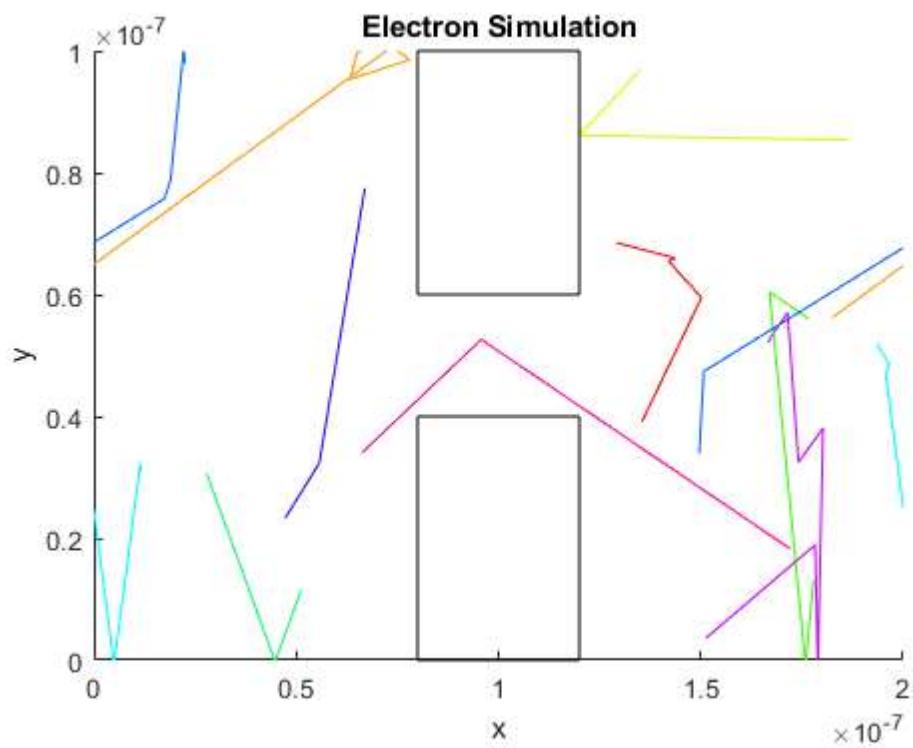


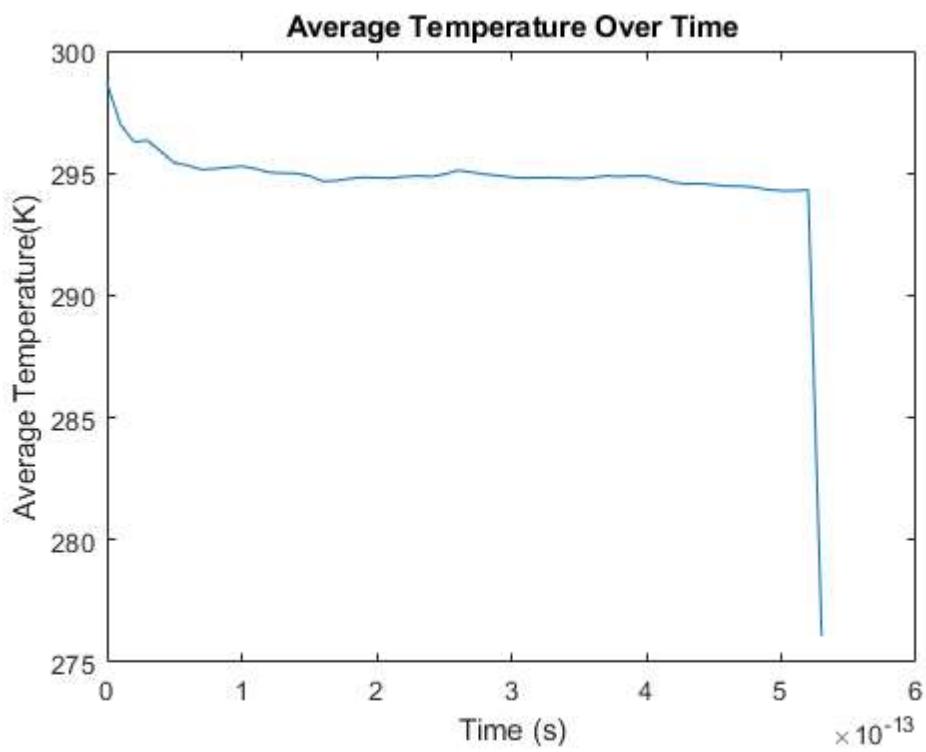
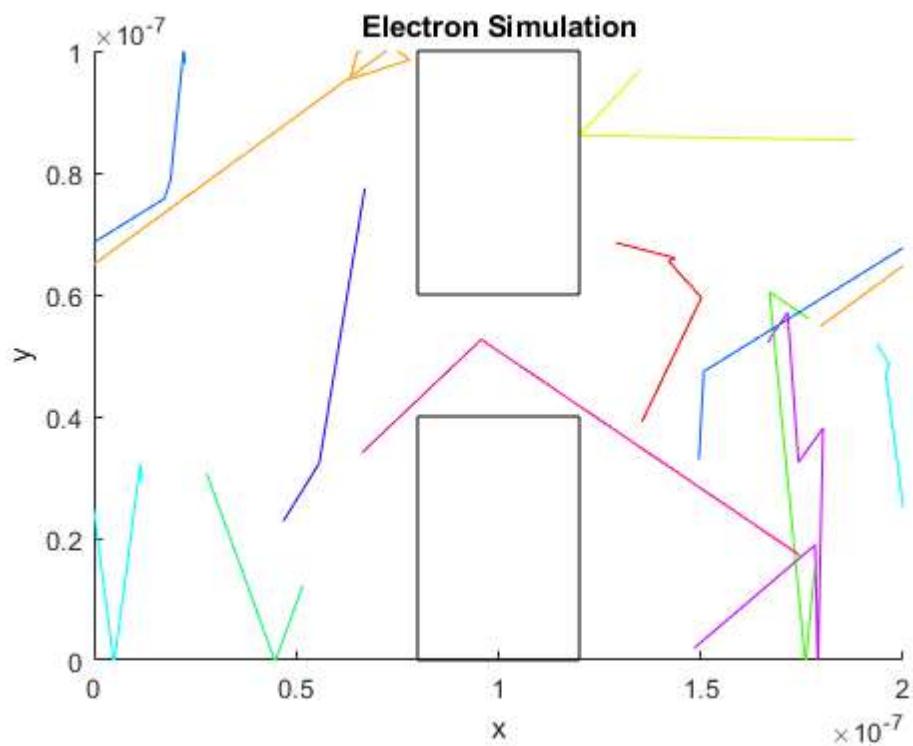


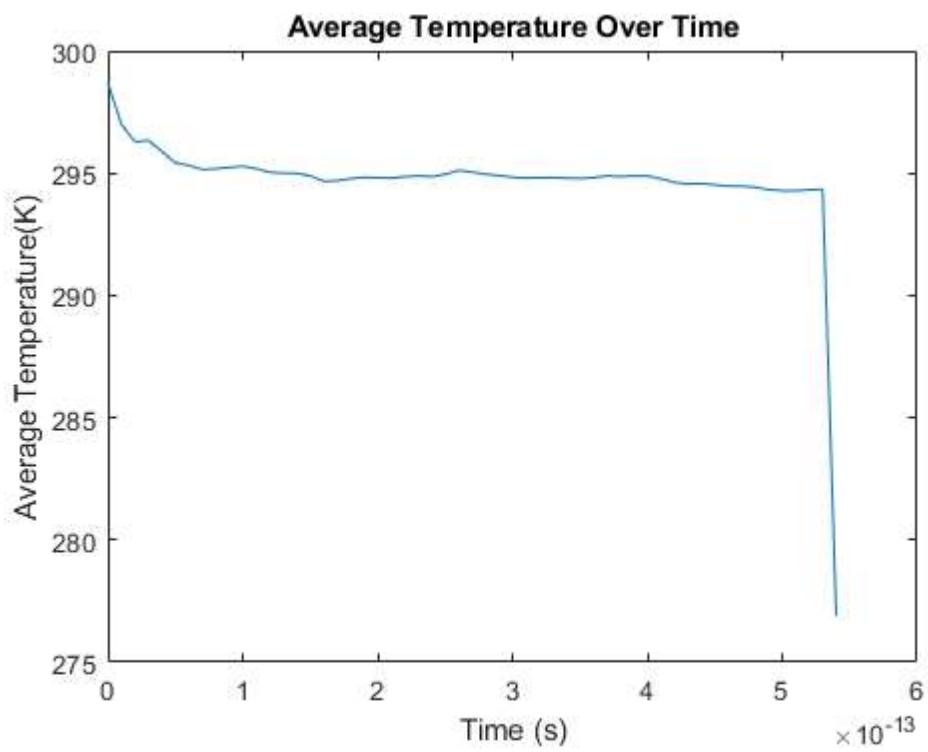
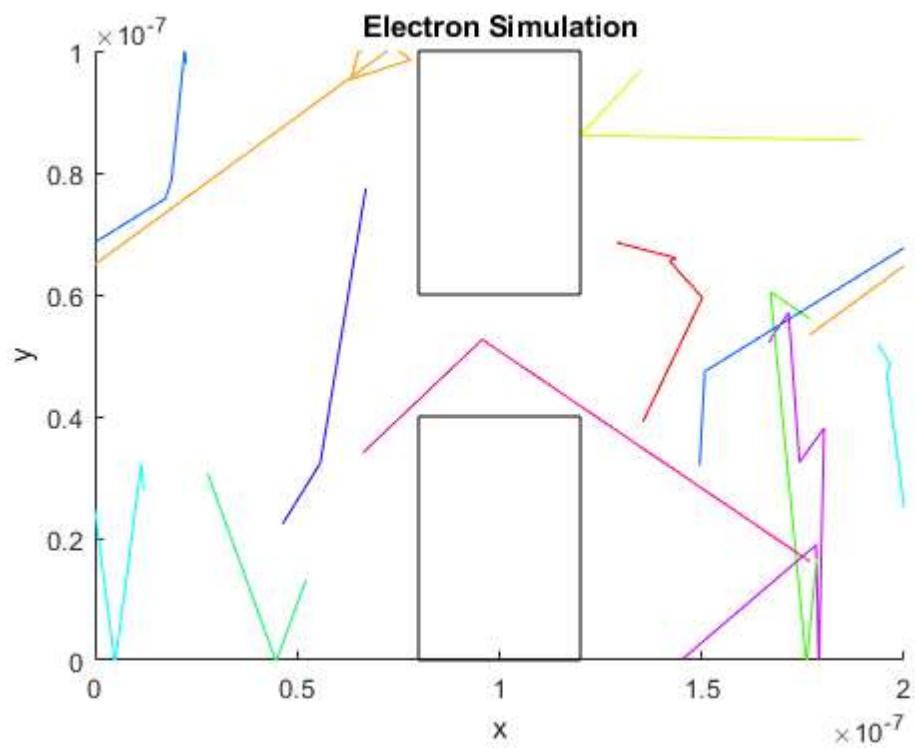


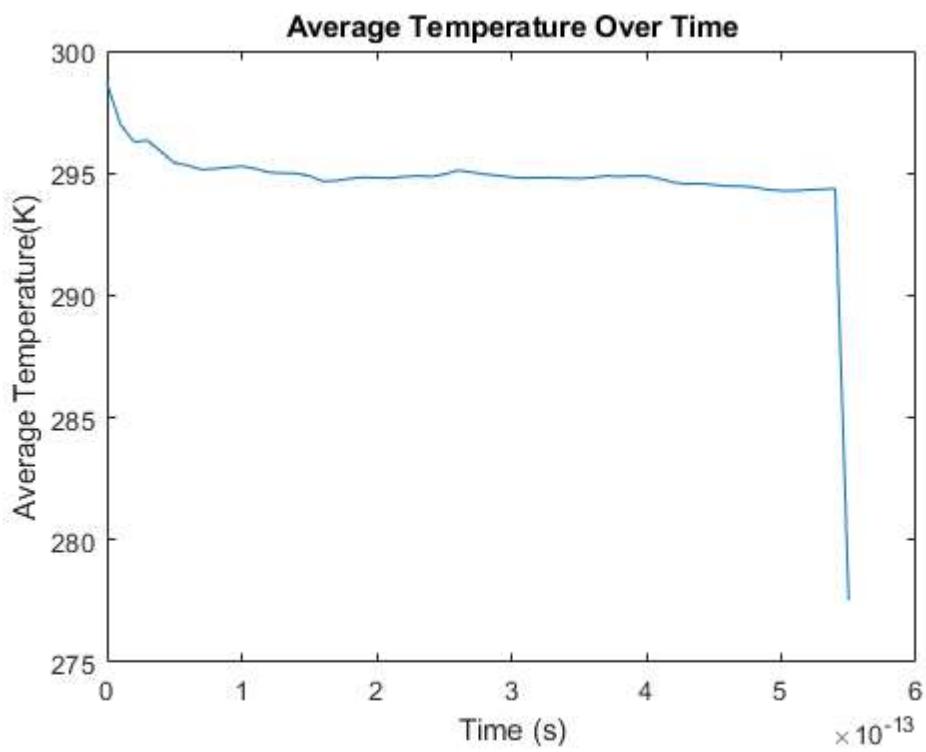
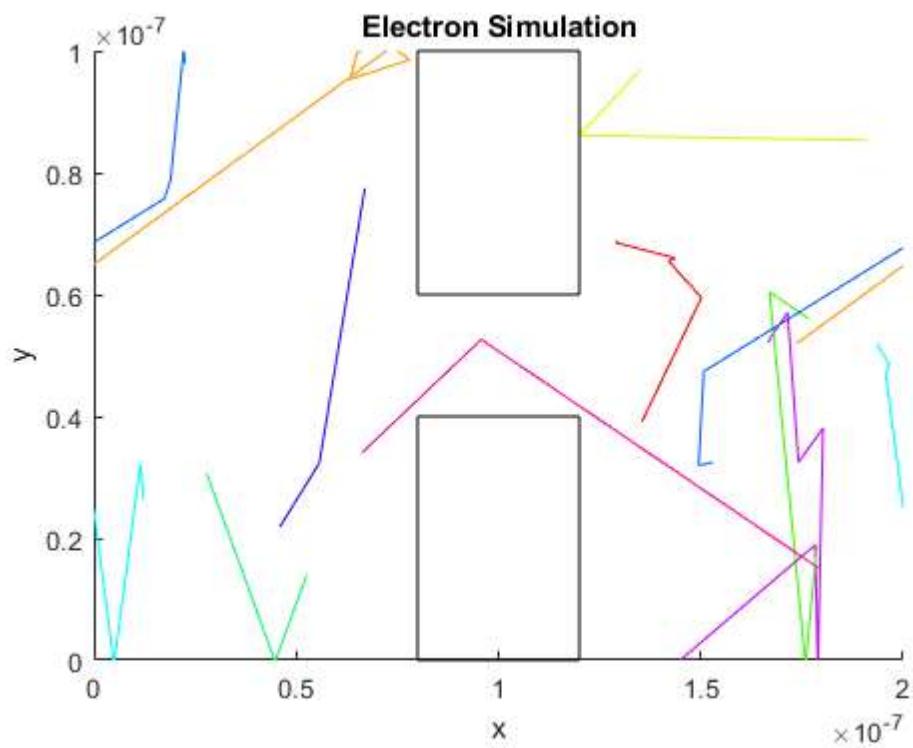


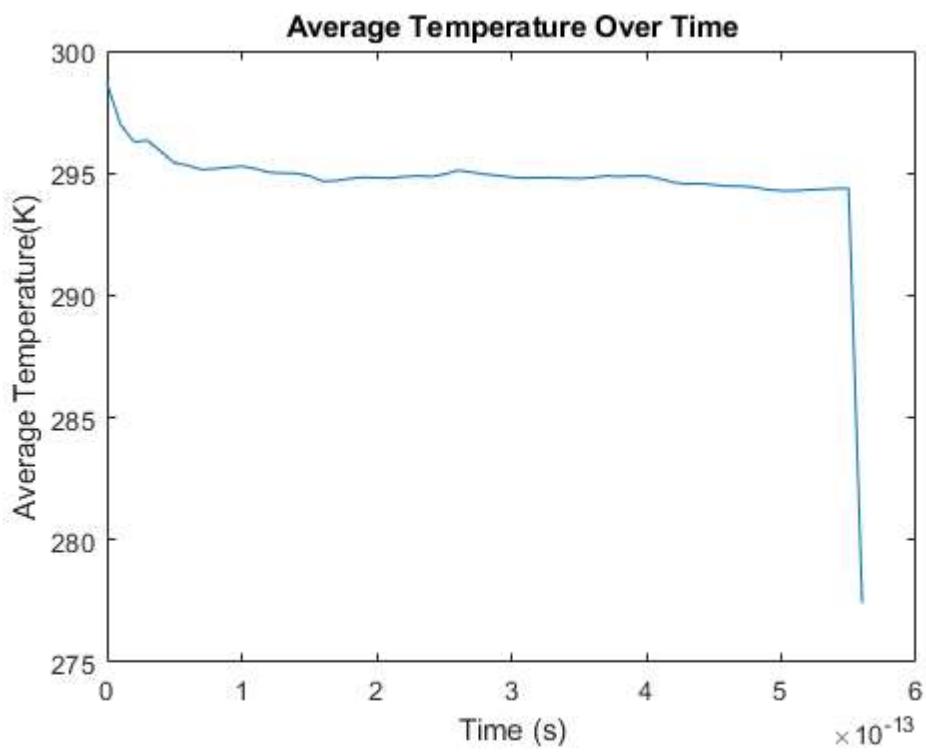
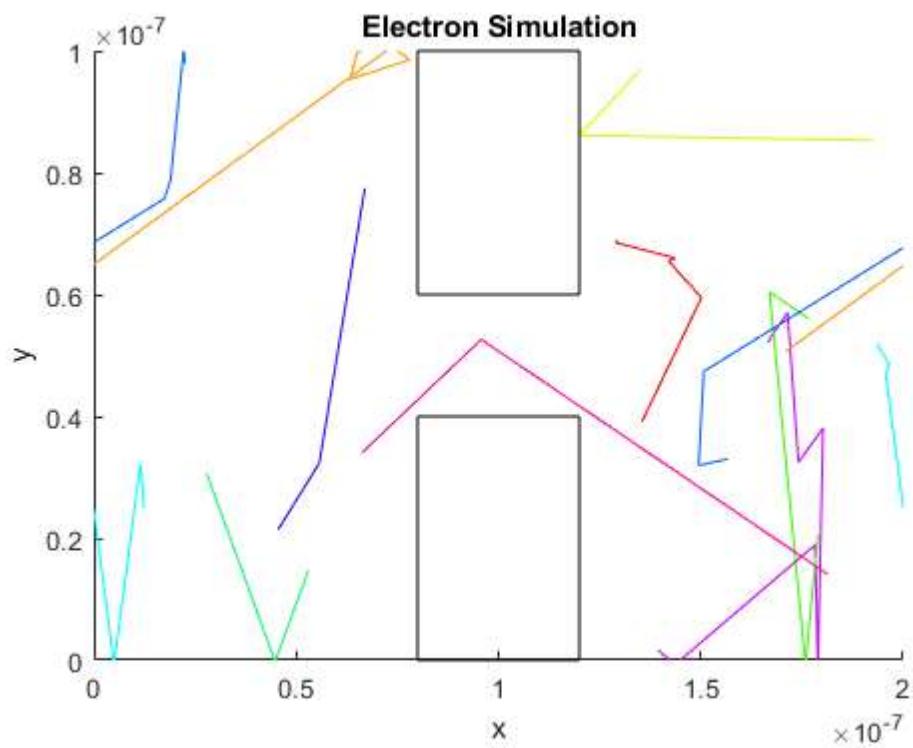


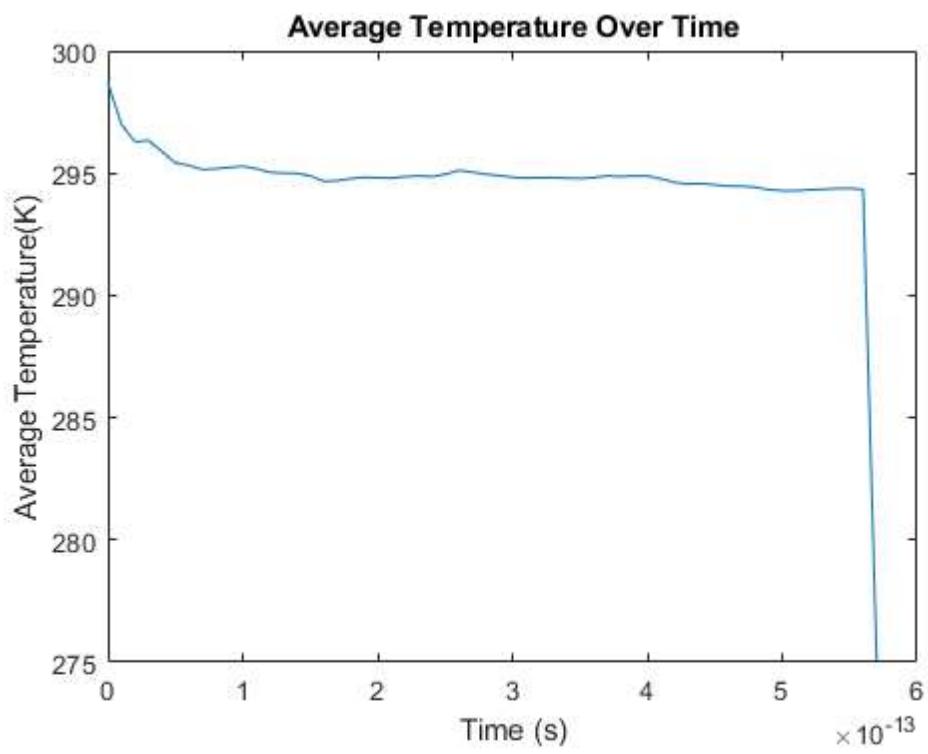
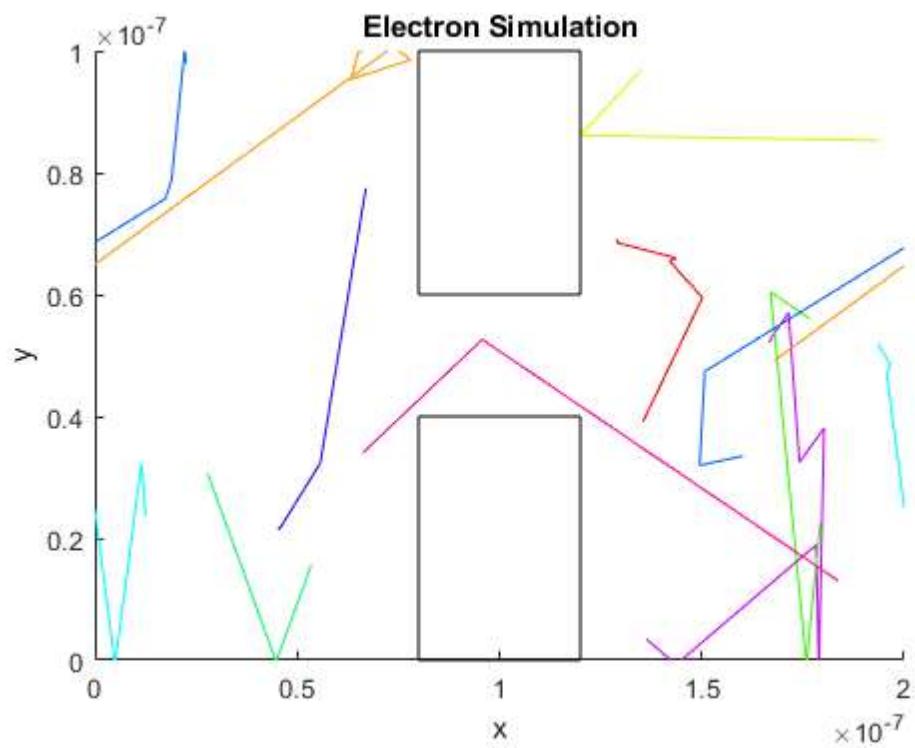


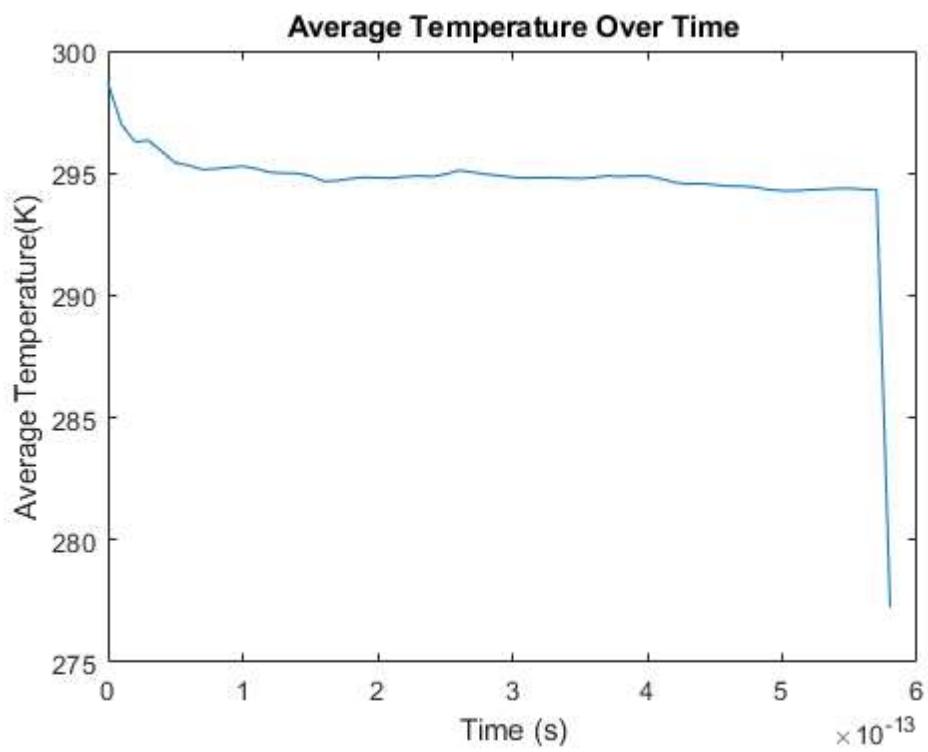
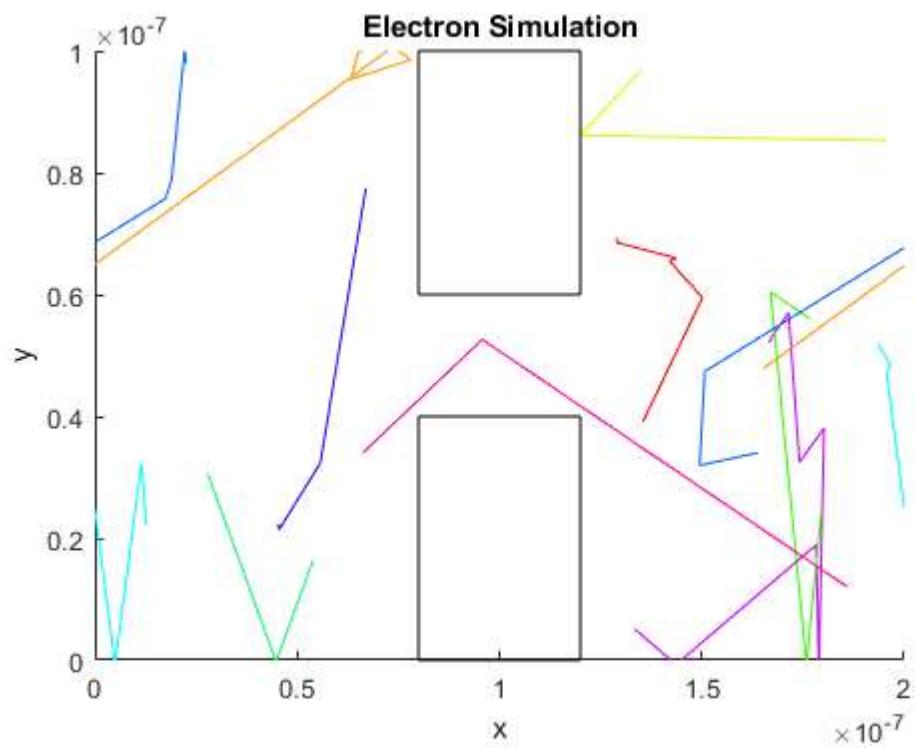


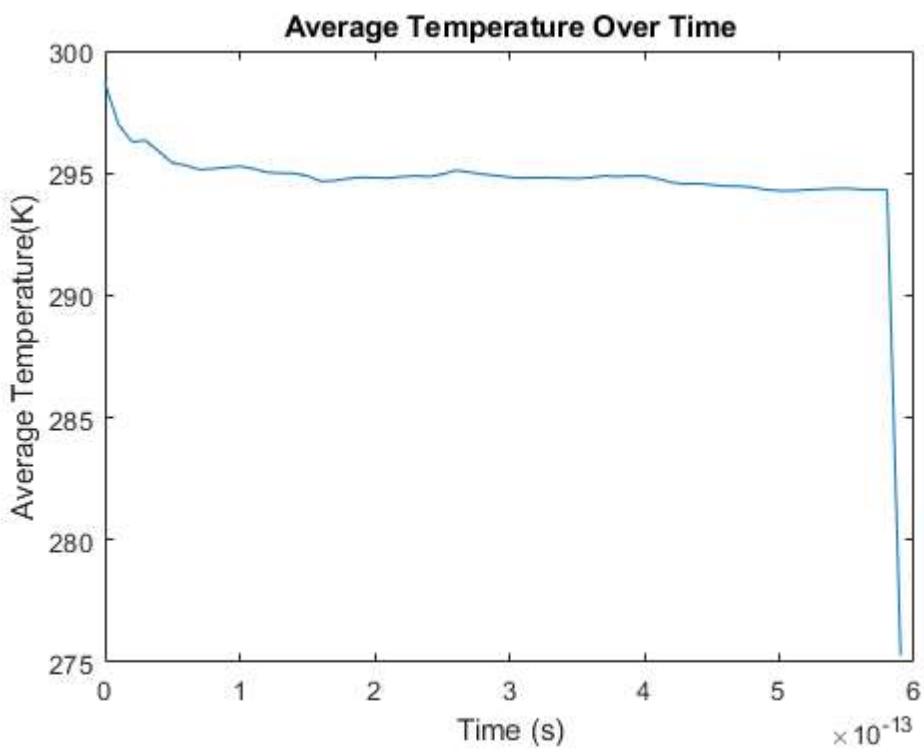
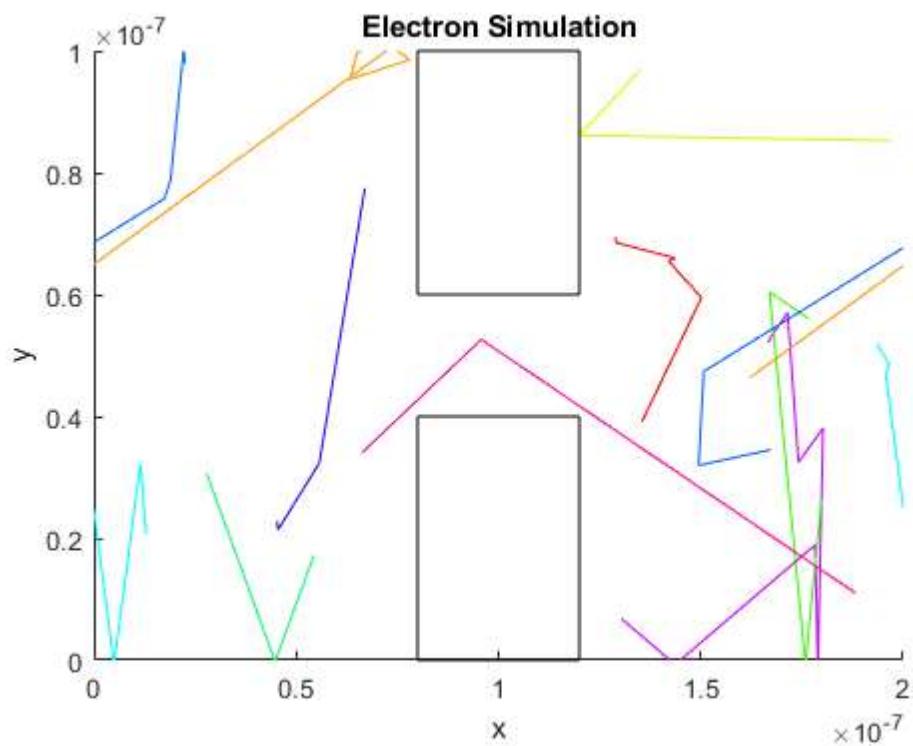


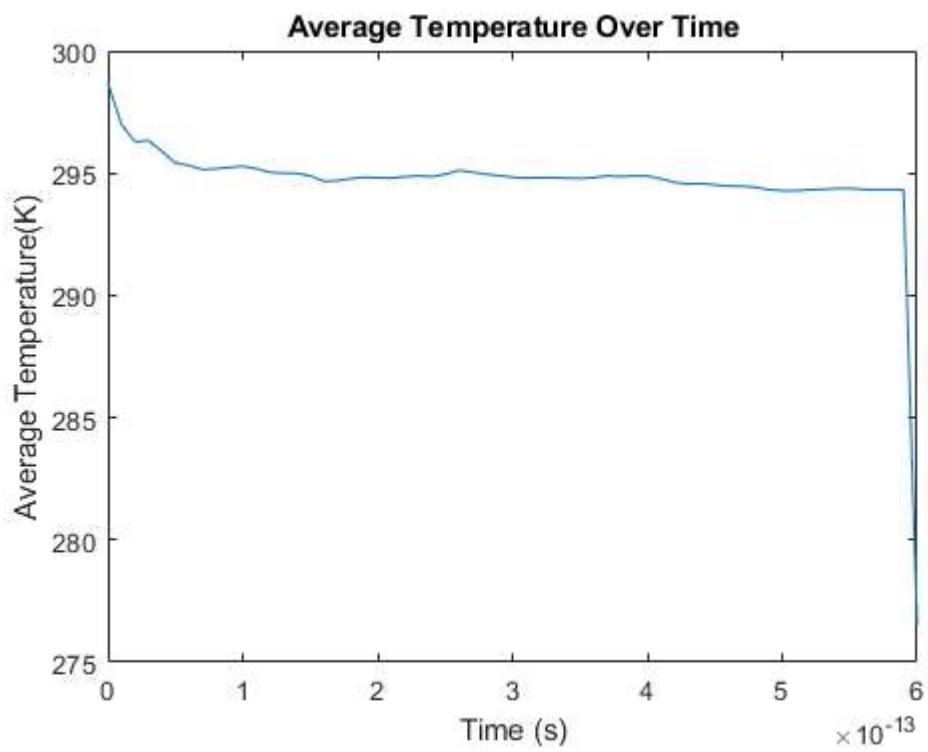
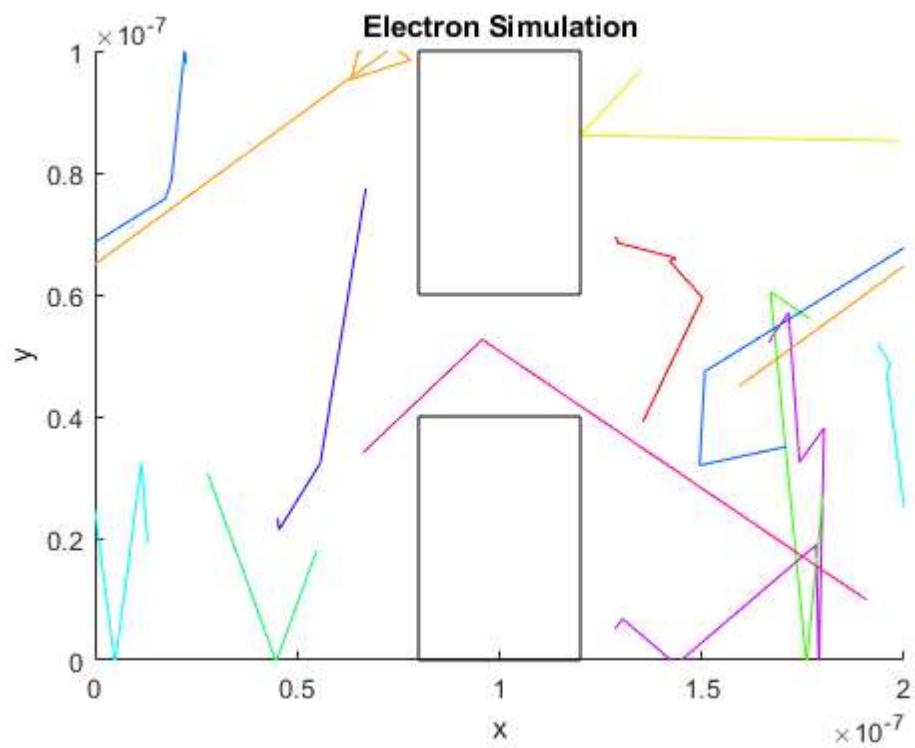


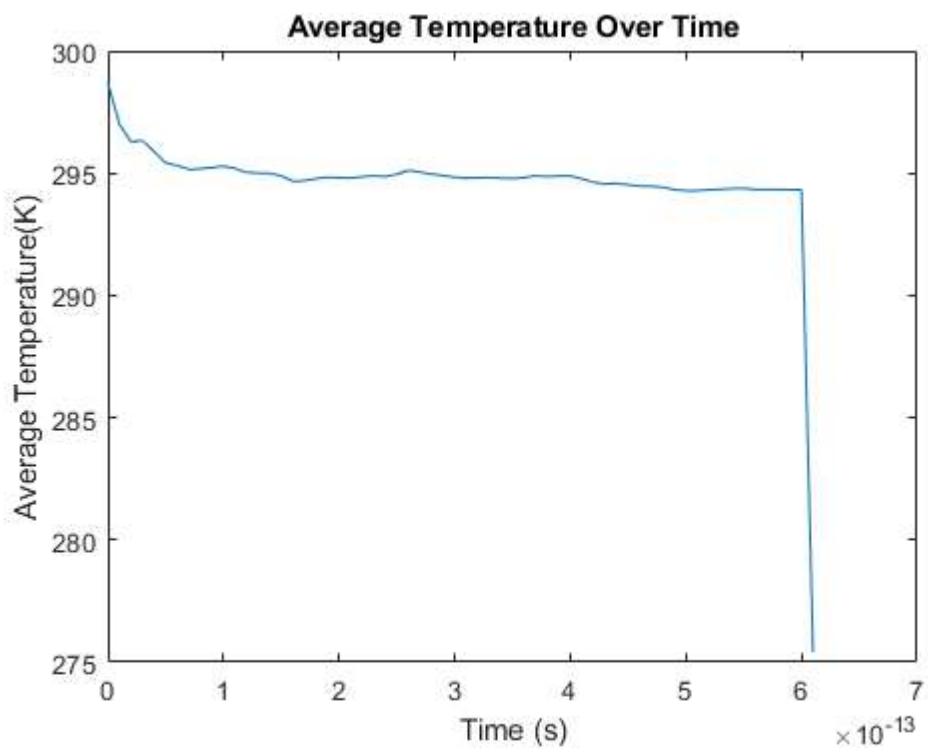
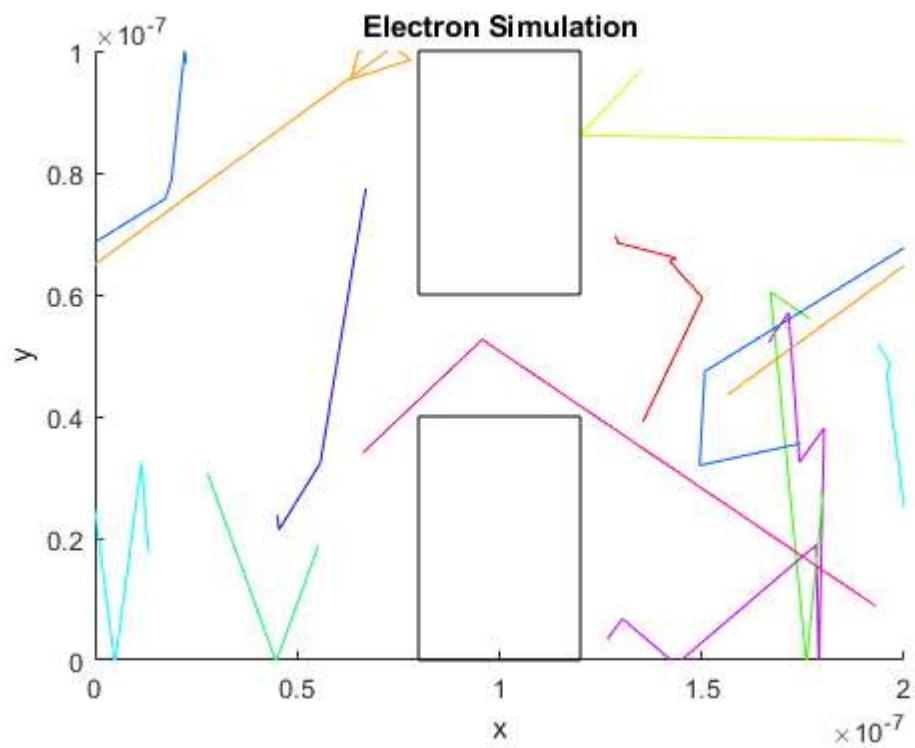


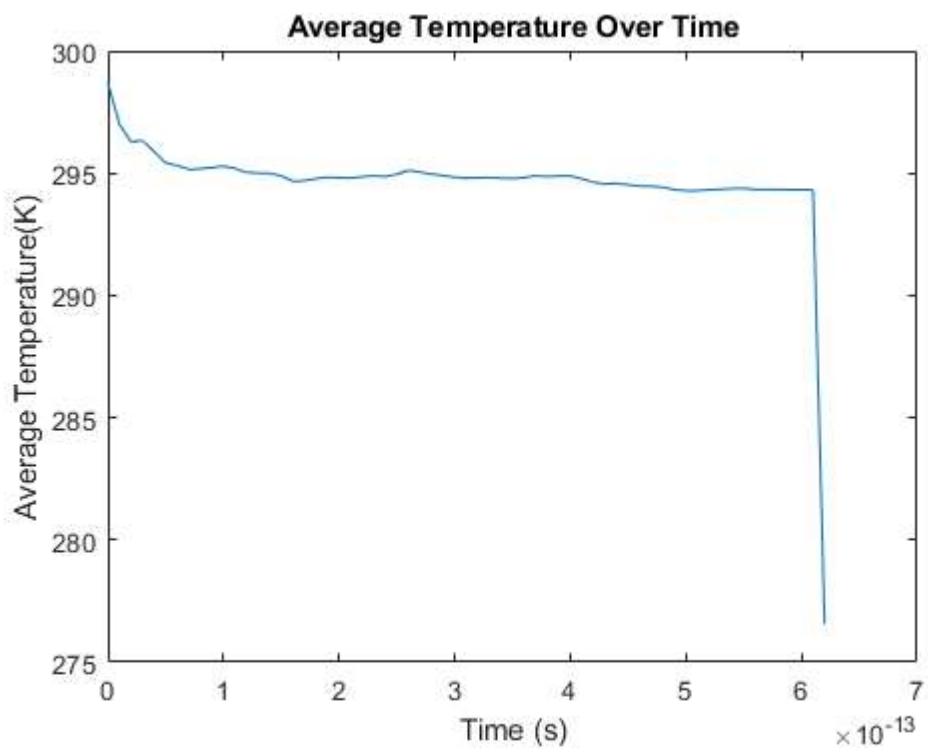
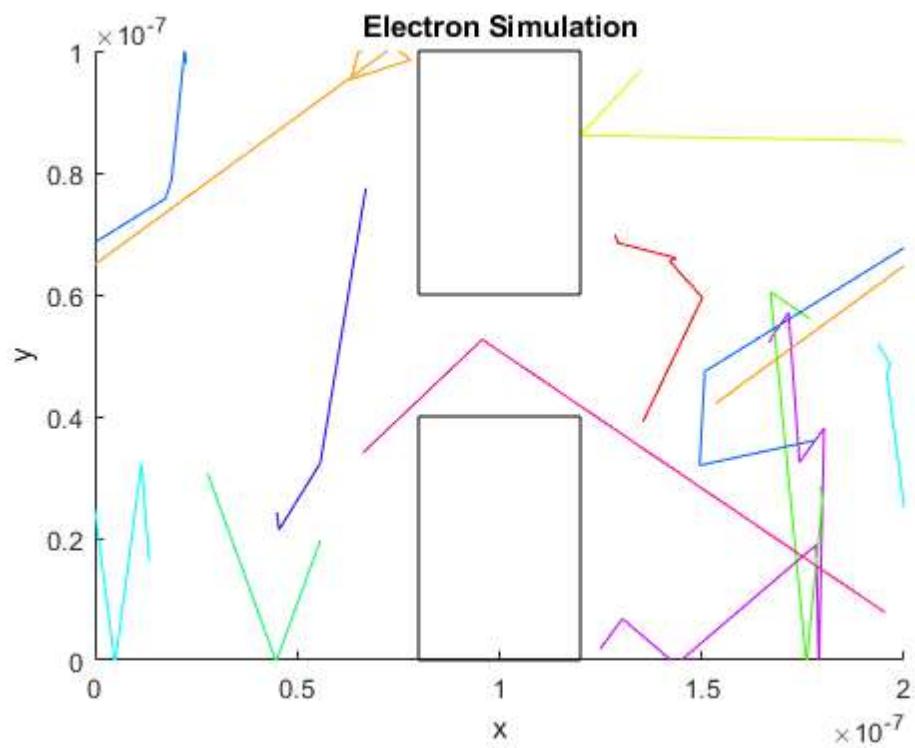


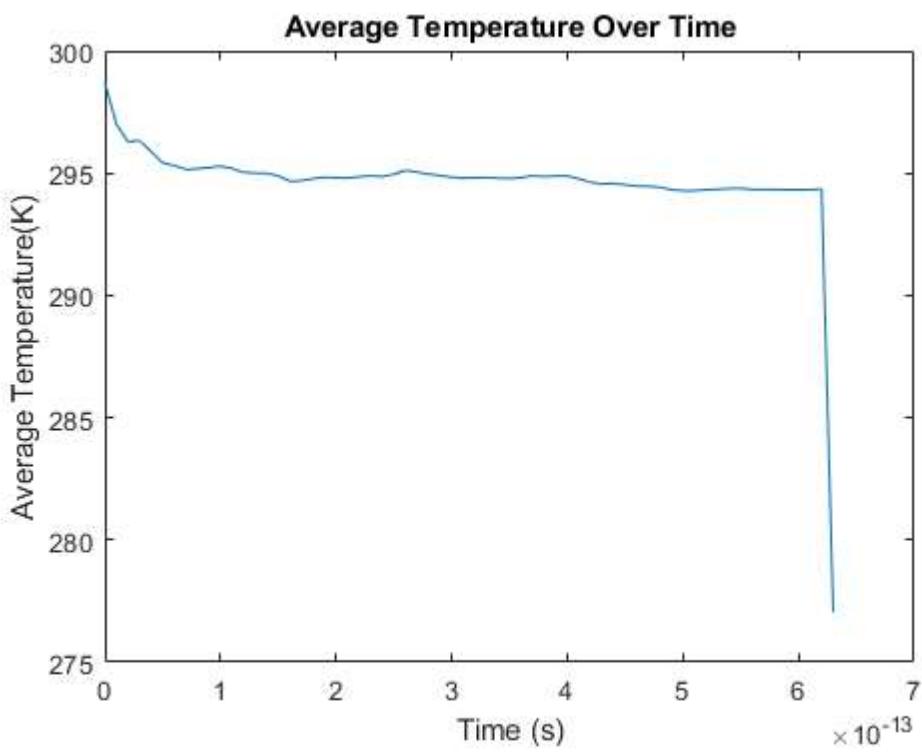
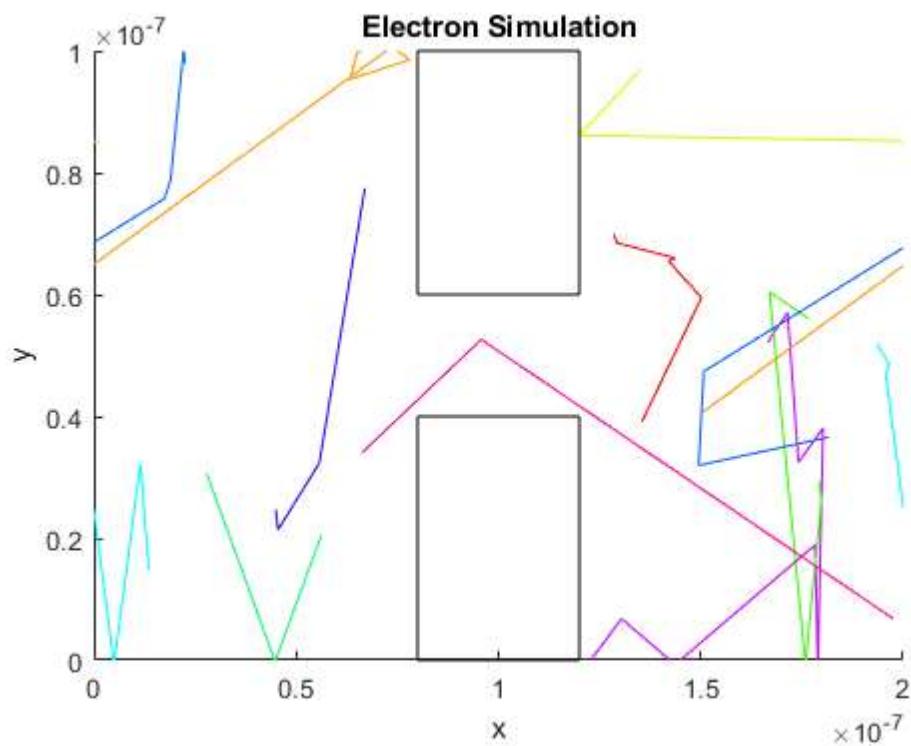


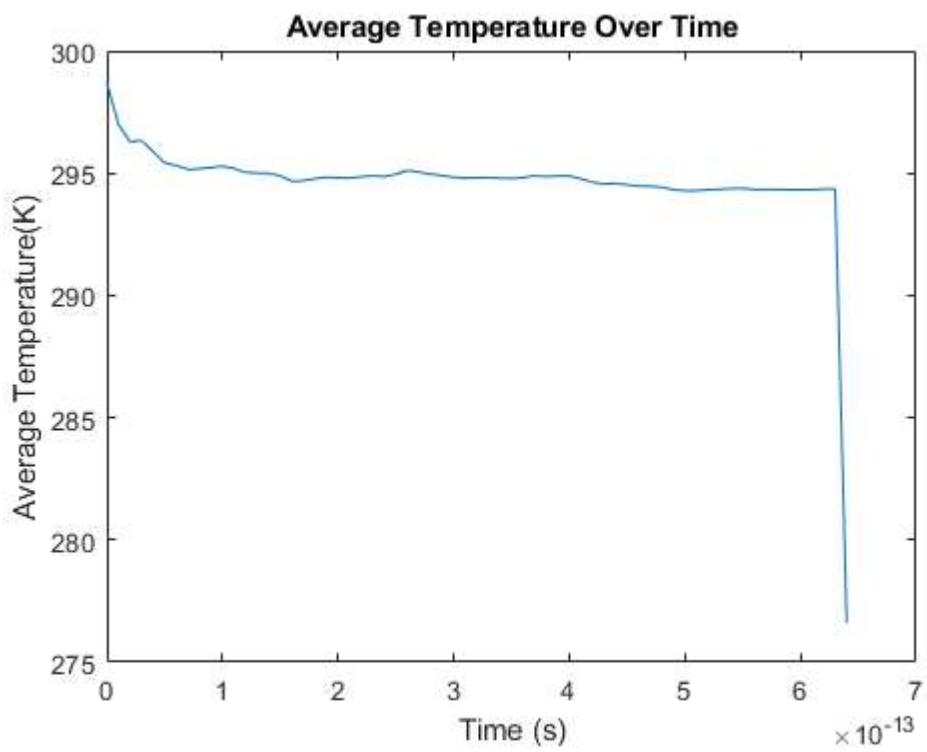
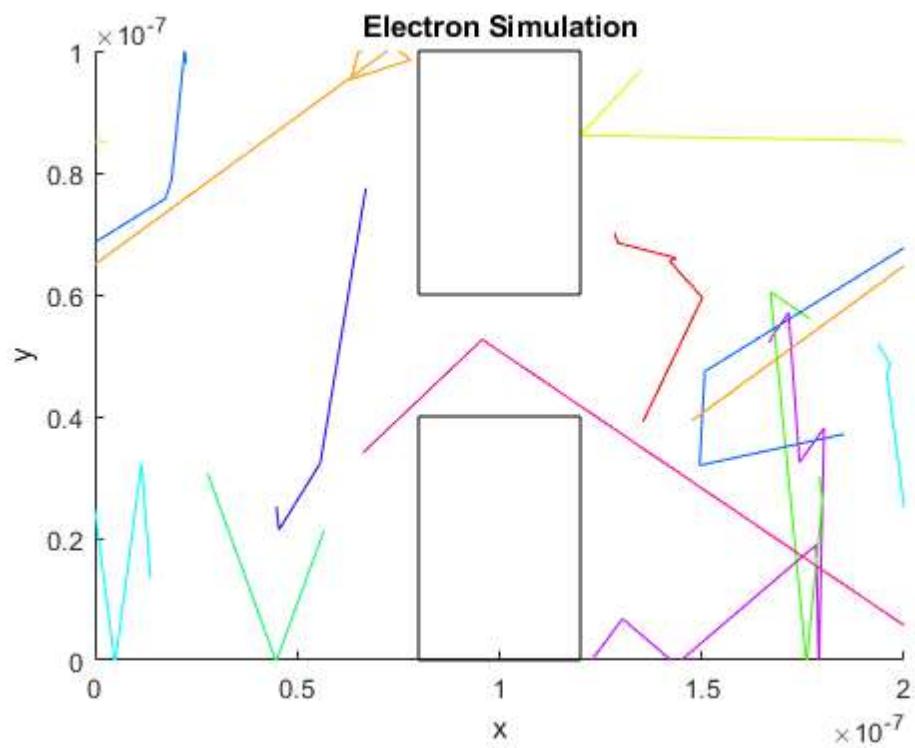


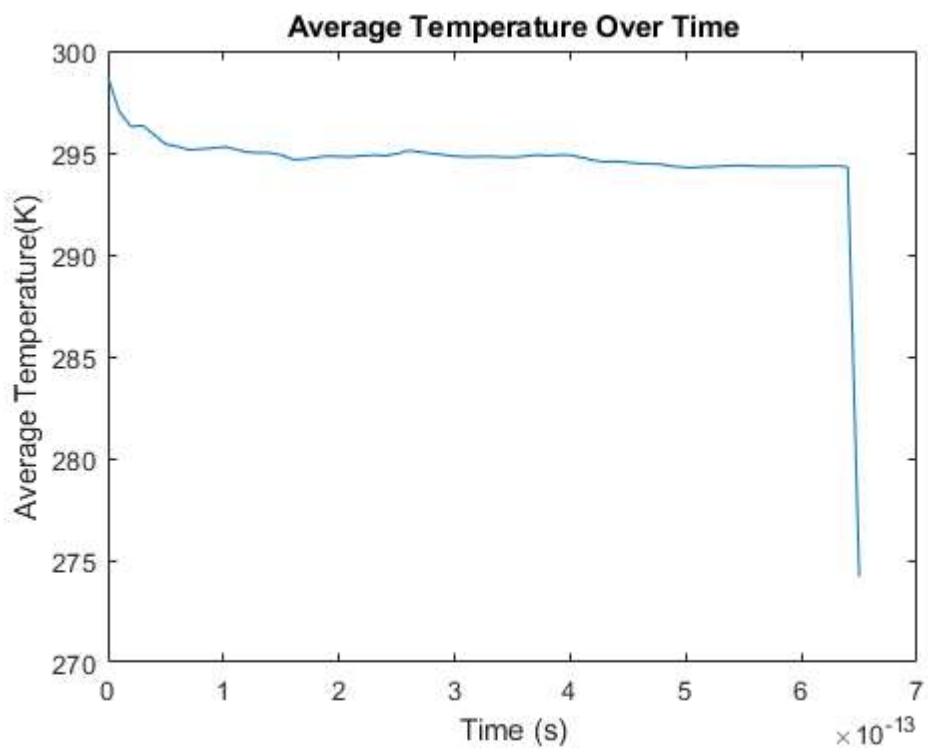
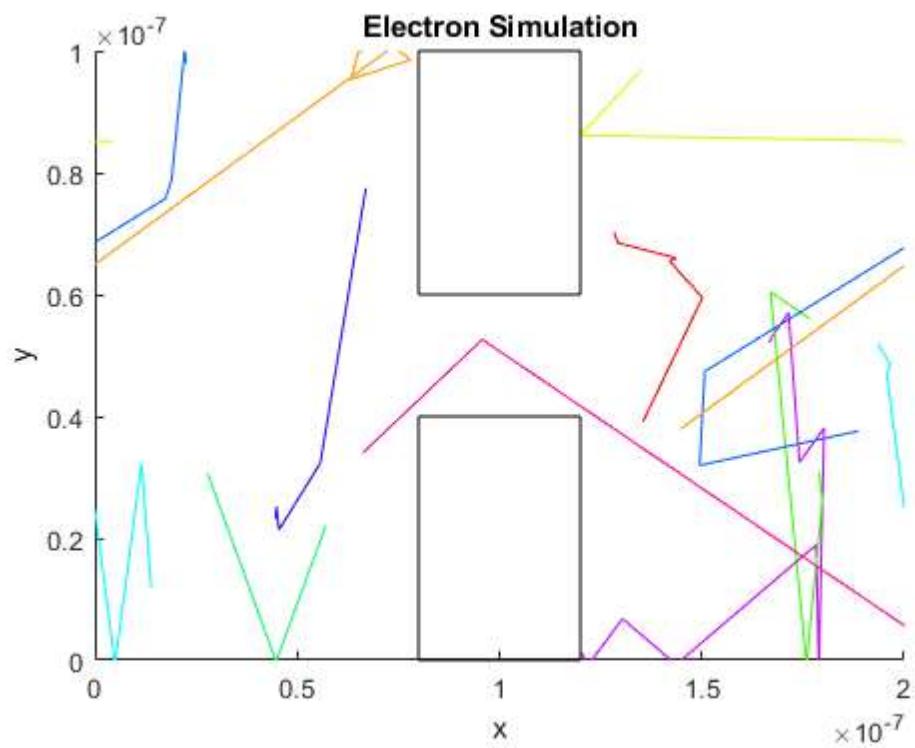


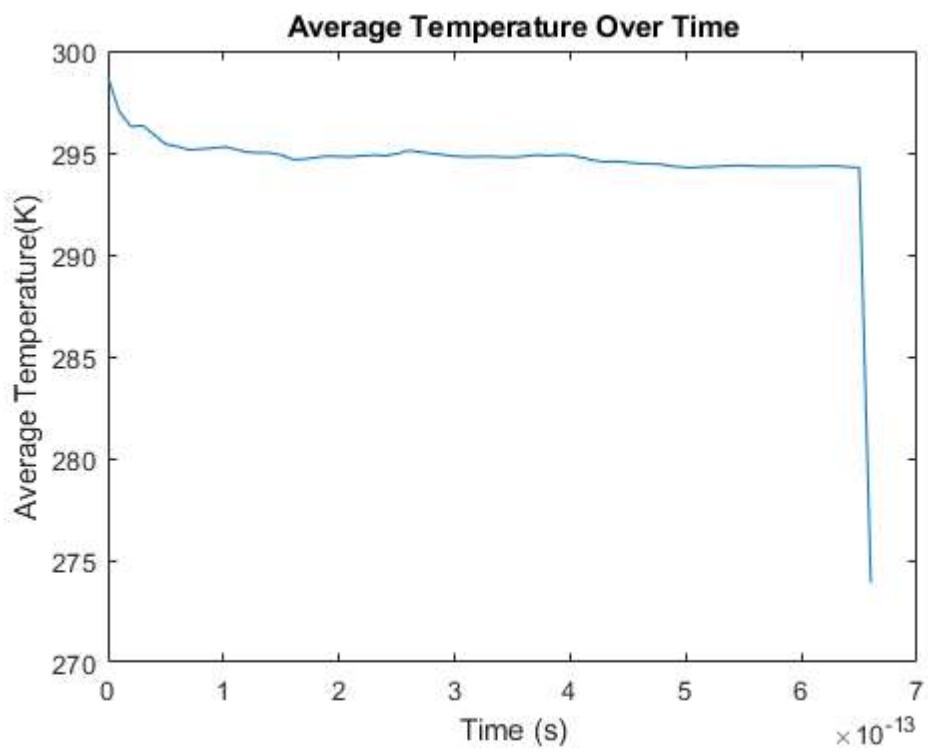
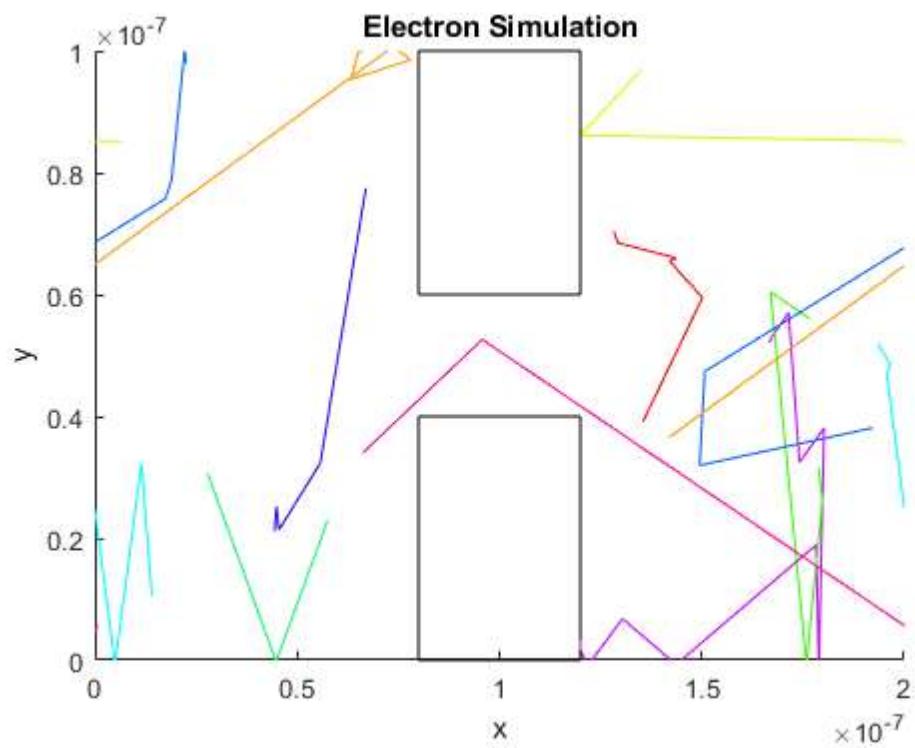


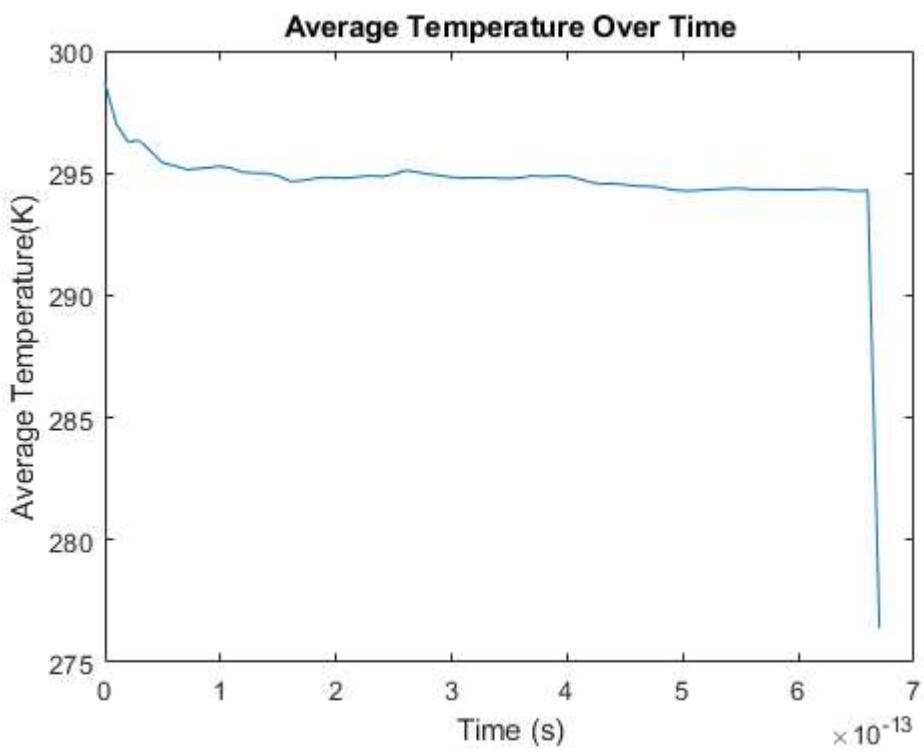
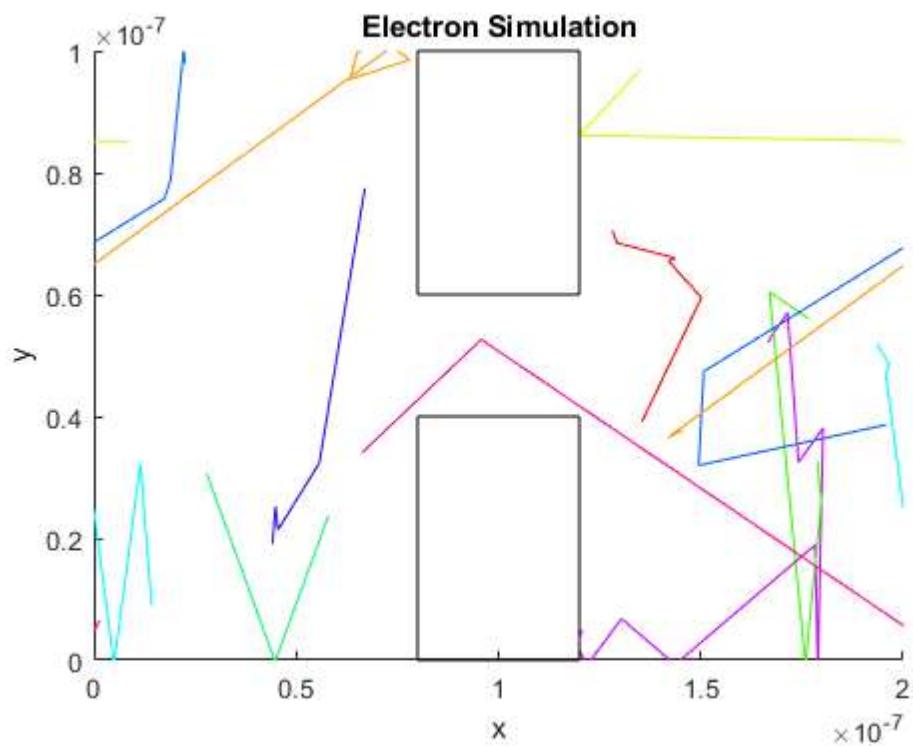


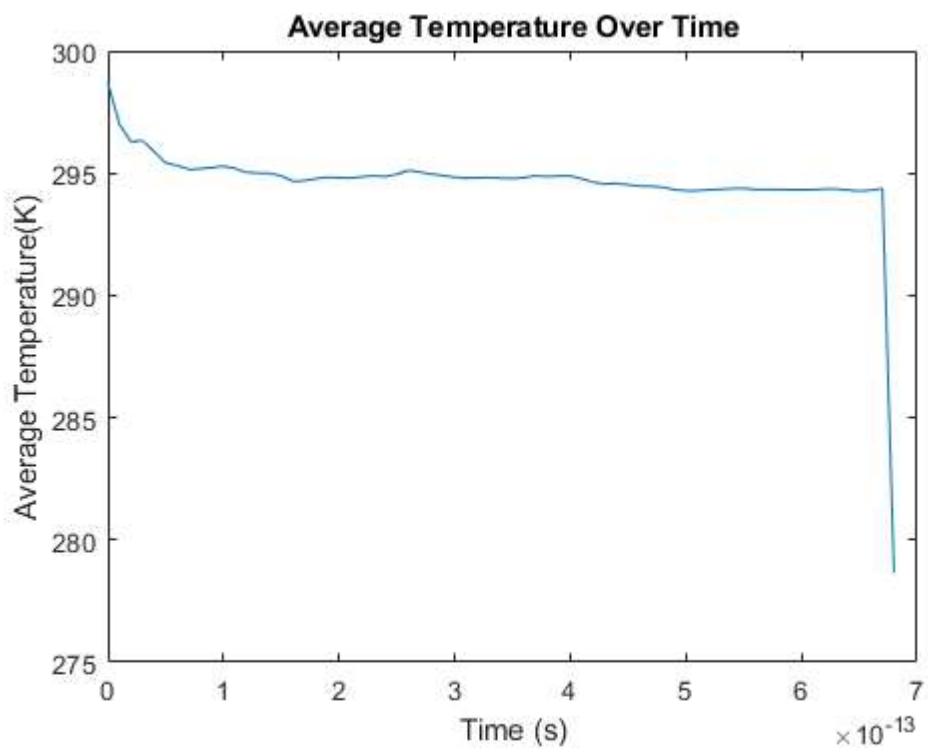
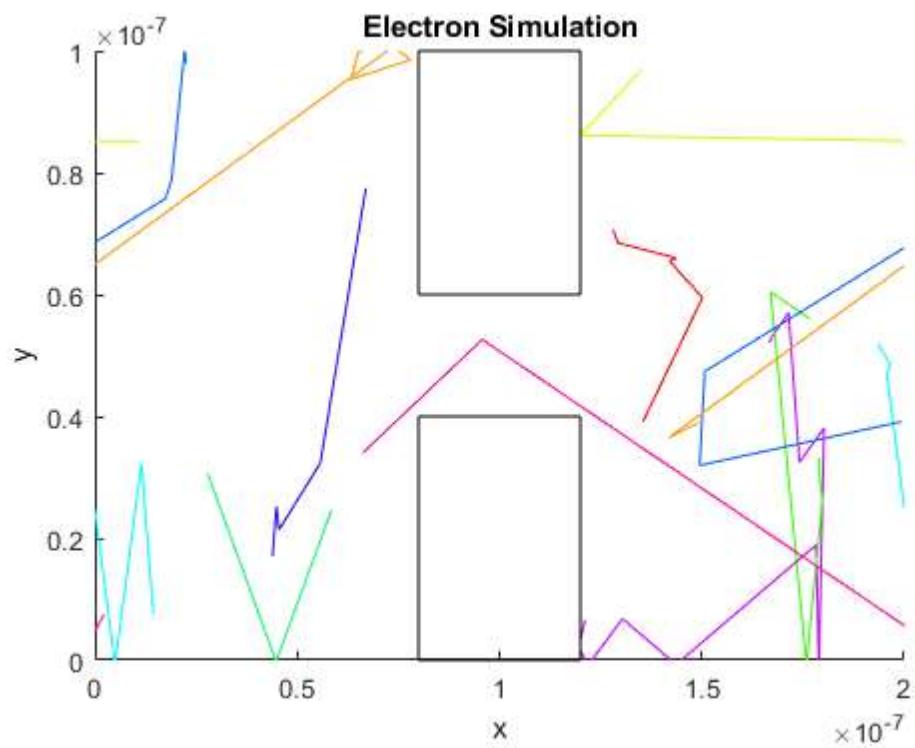


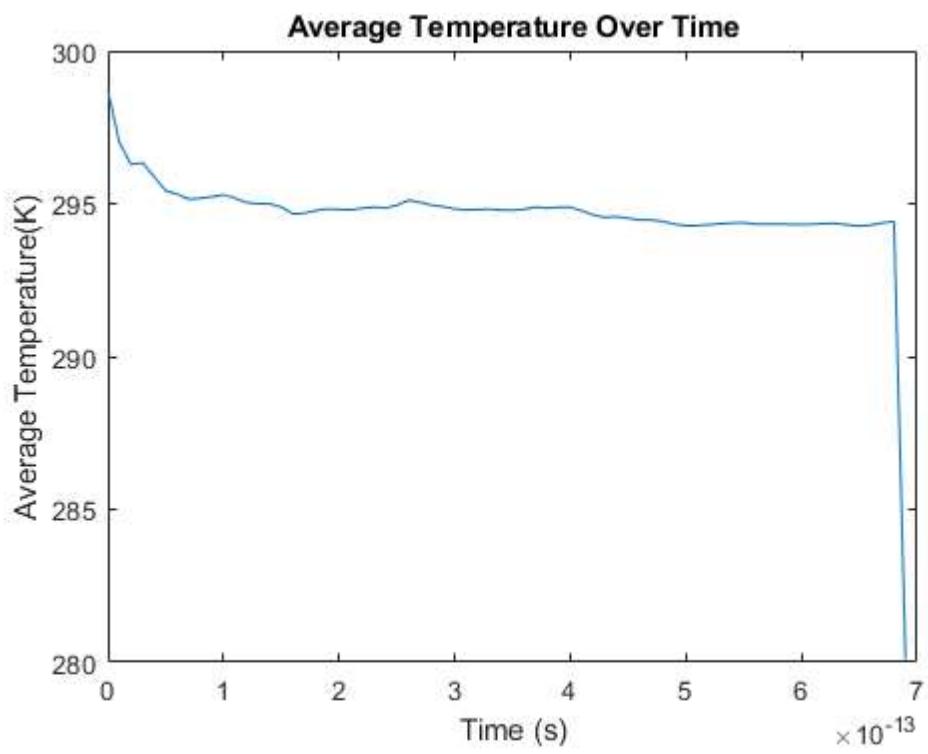
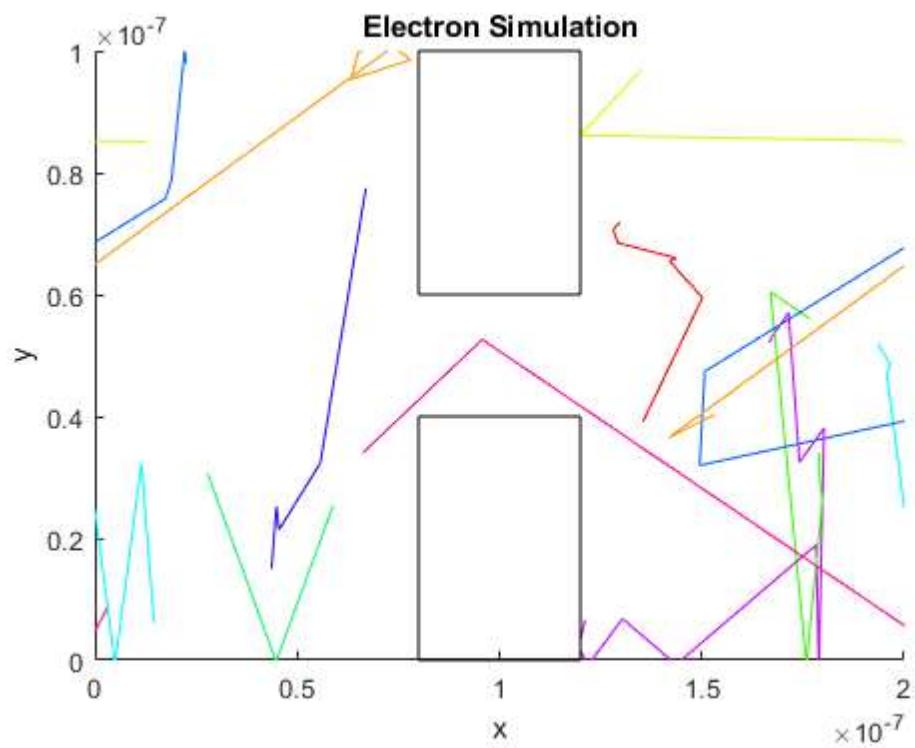


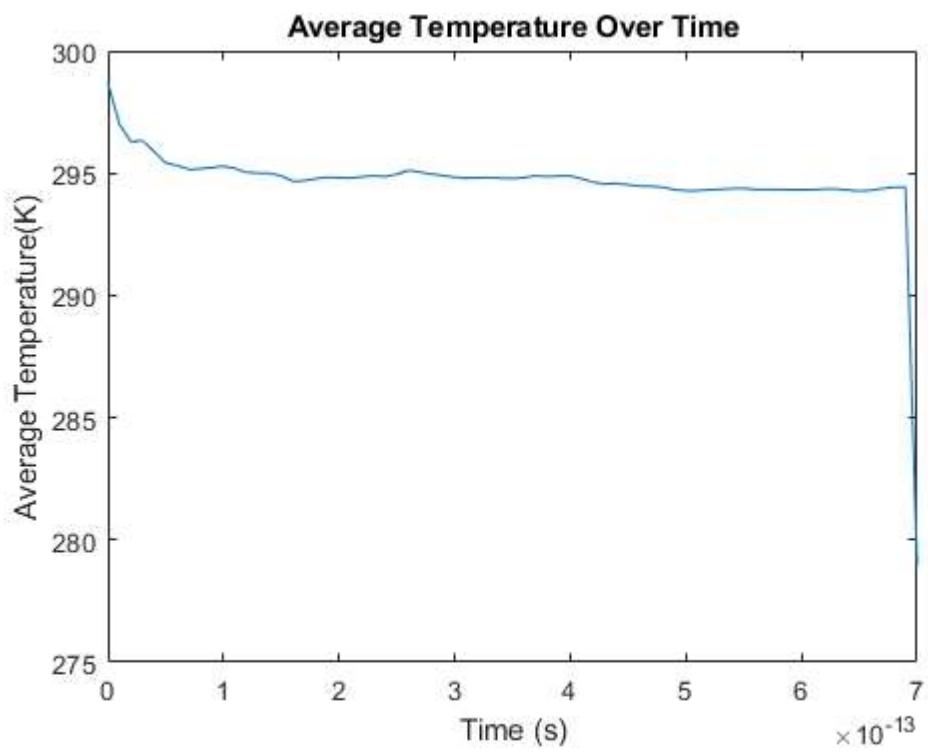
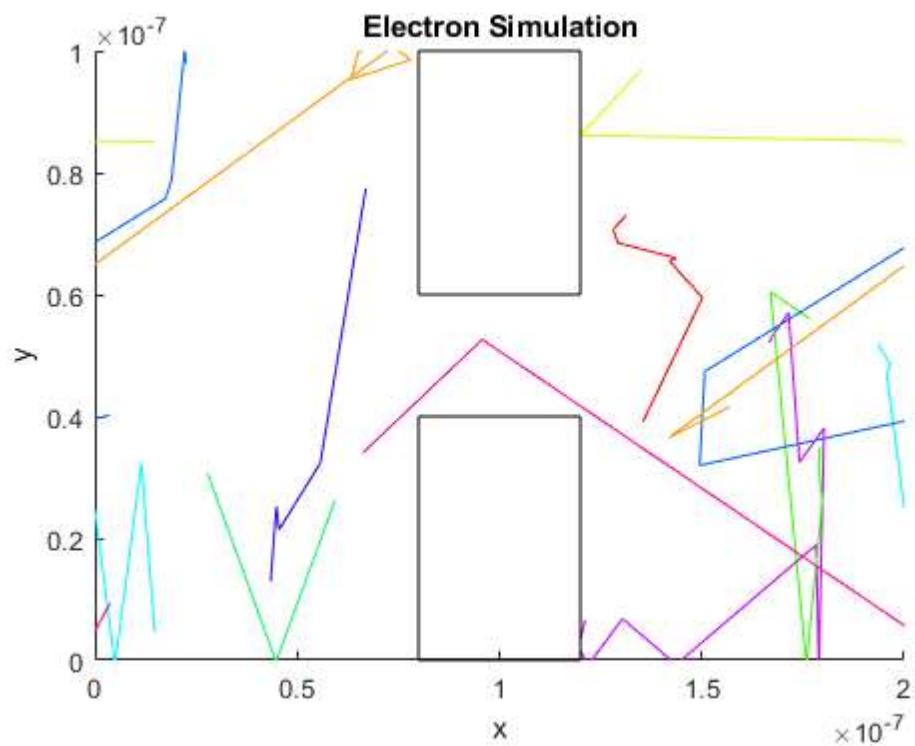


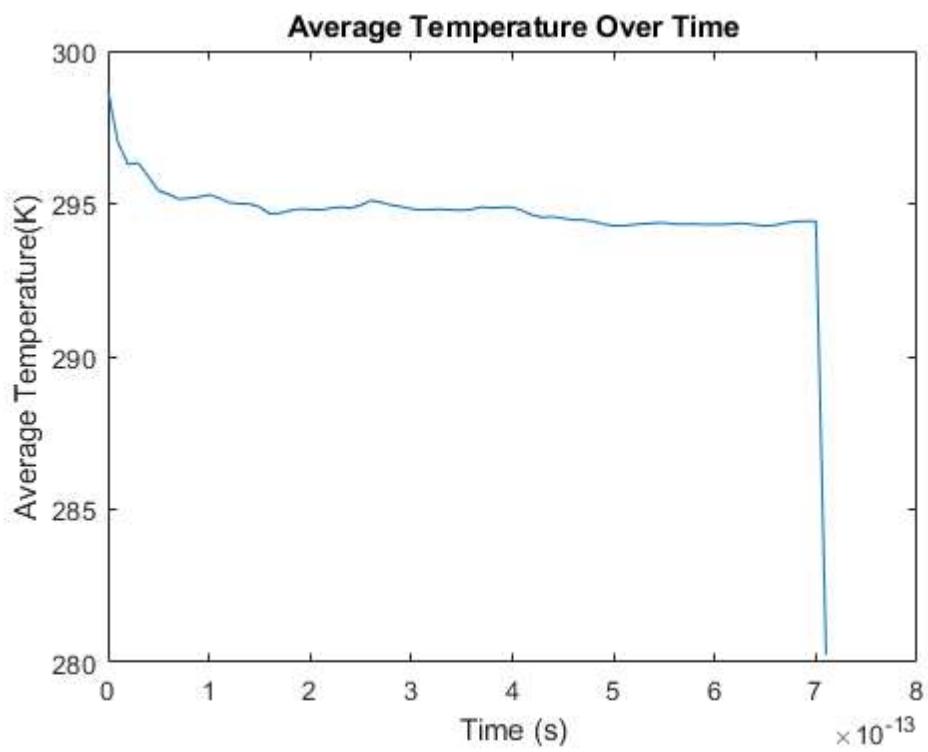
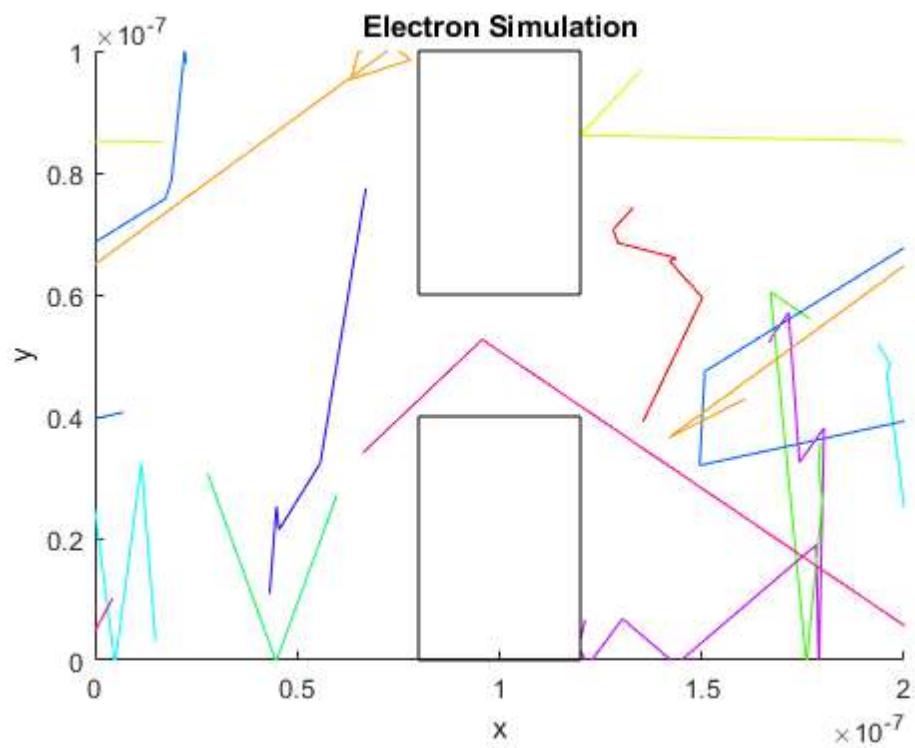


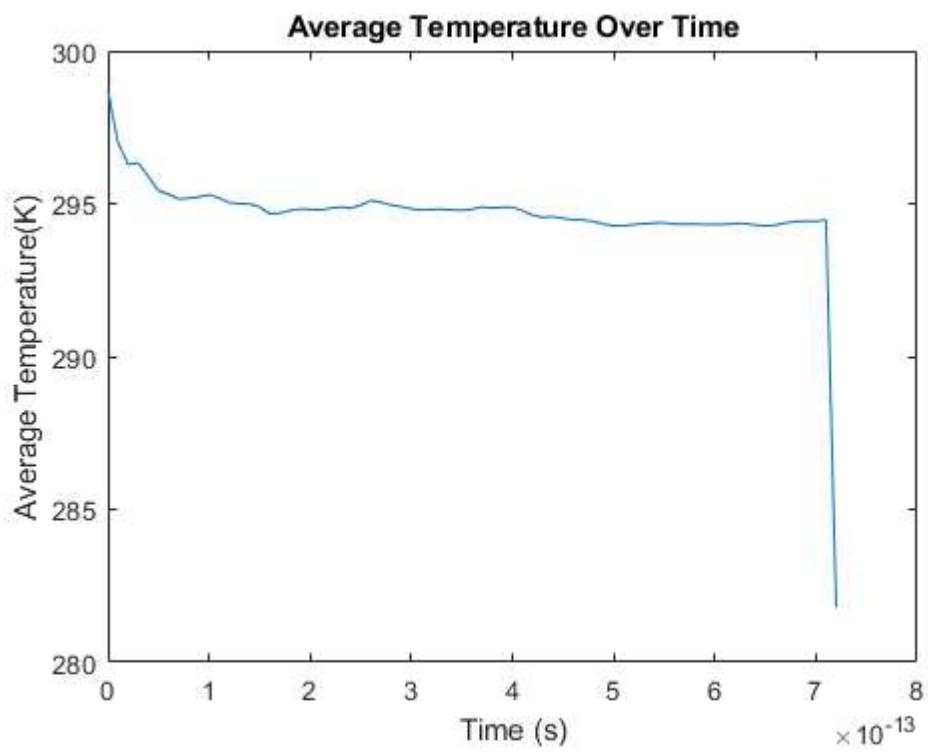
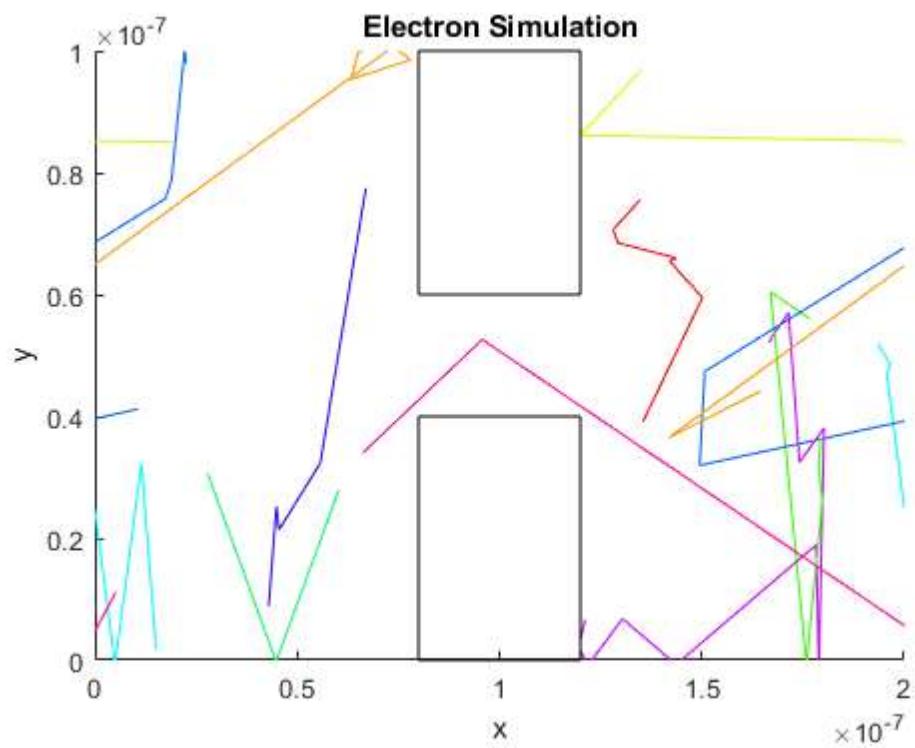


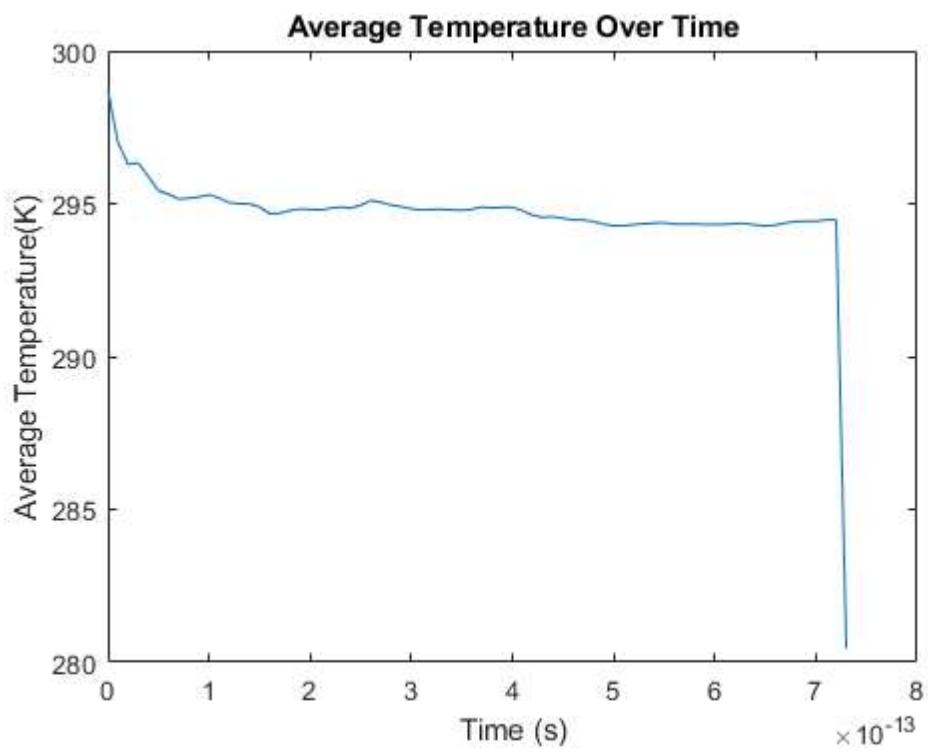
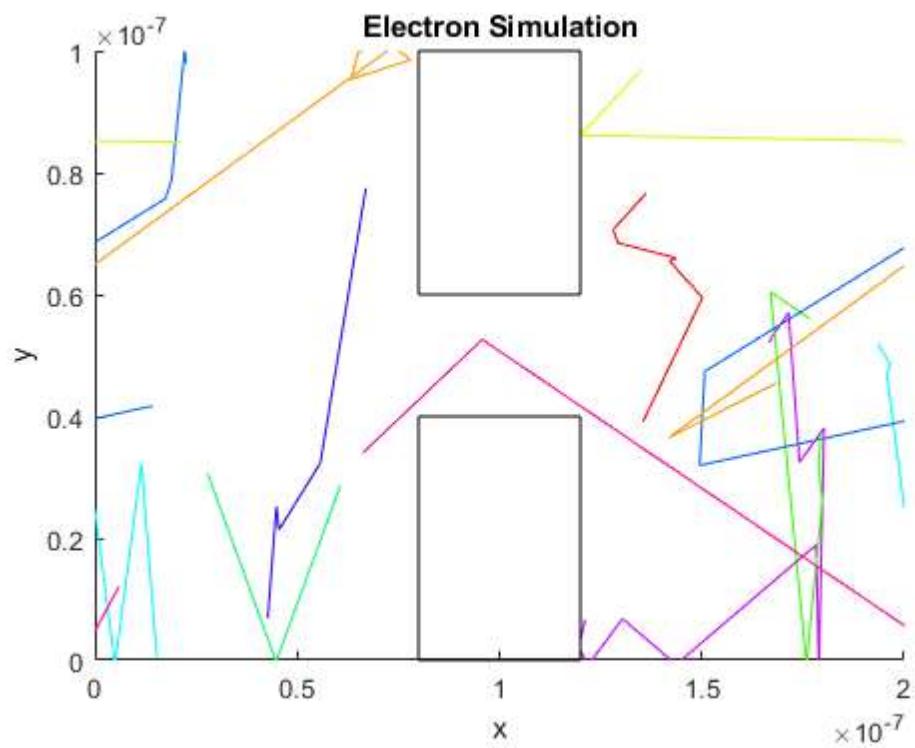


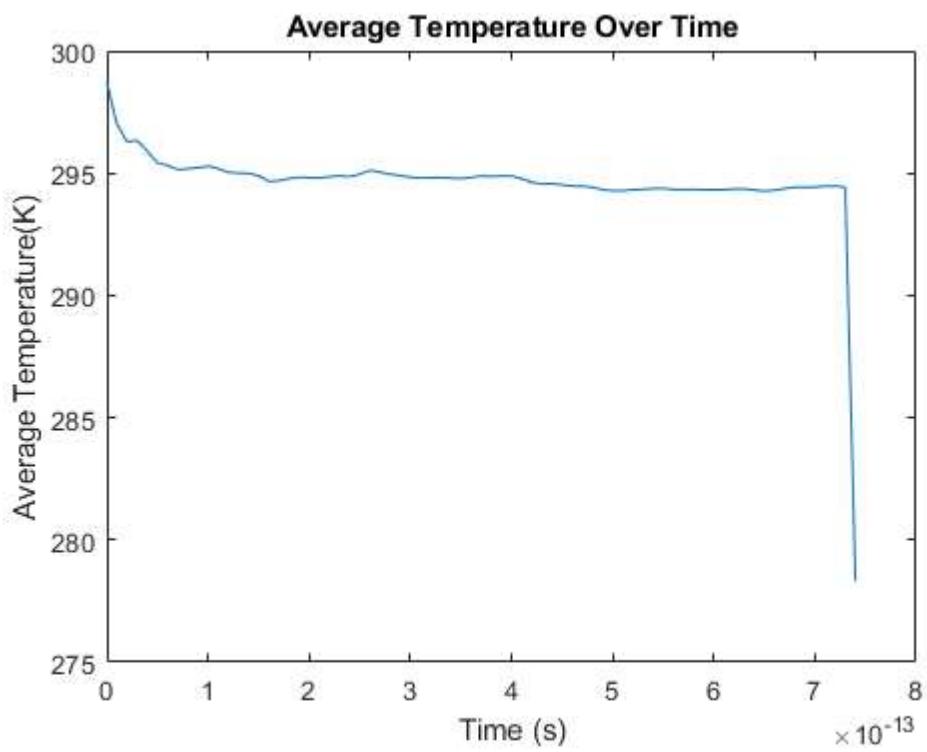
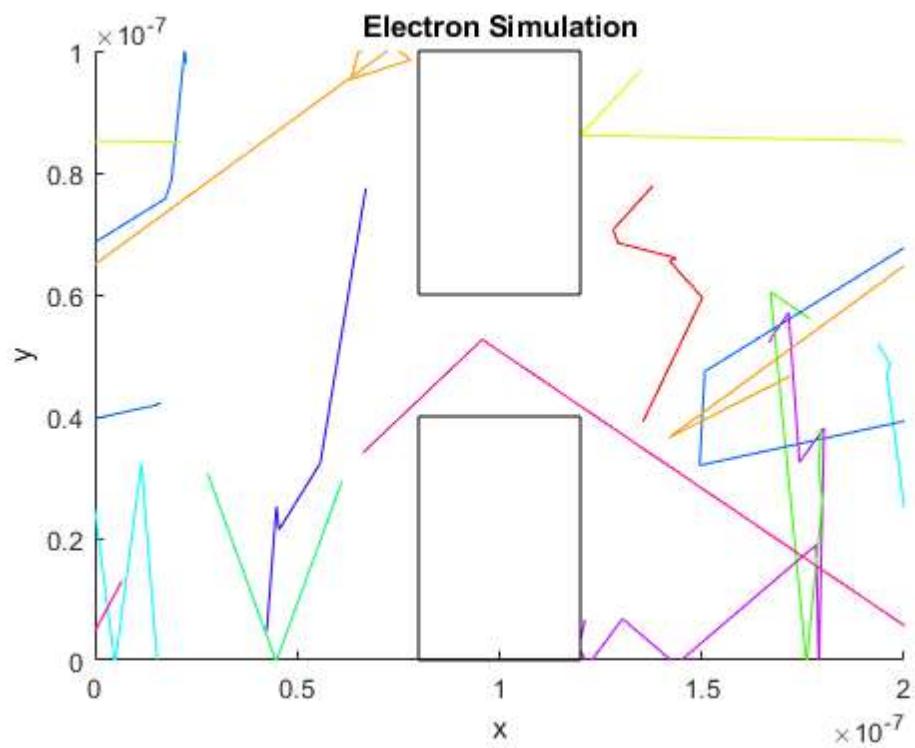


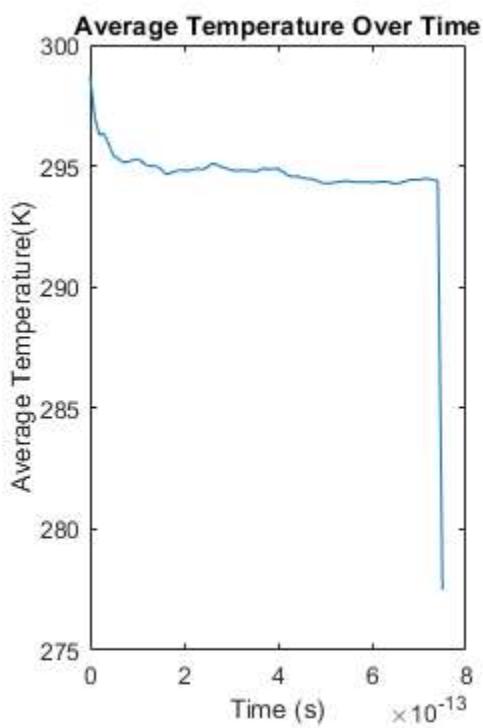
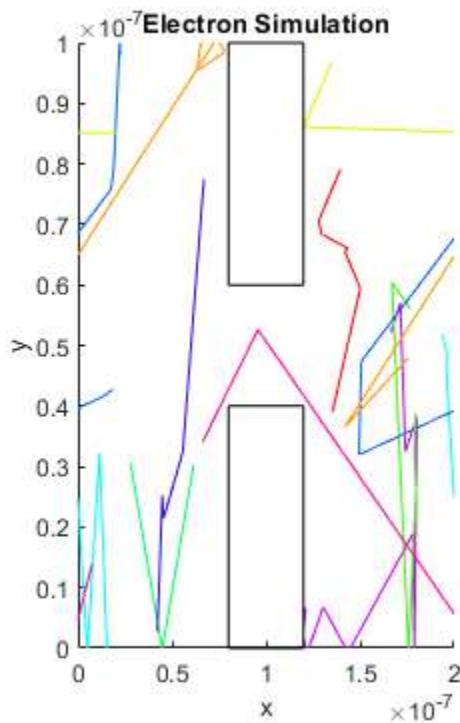












Next, the previous x and y values are updated with the current values of x and y.

```
xp = x;
yp = y;
```

Scattering is done, pscat is the probability to scatter, then scat represents the indices which must scatter.

```
pscat = 1 - exp(-dt/tmn);
scat = pscat > rand(NE, 1);

scatlength = length(vx(scat));
```

Next, the velocities are scattered.

```
if(scatlength>0)
    vx(scat) = 1*sqrt((kB*T)/mn)*randn(scatlength, 1);%velocity x and y
    vy(scat) = 1*sqrt((kB*T)/mn)*randn(scatlength, 1);
end
```

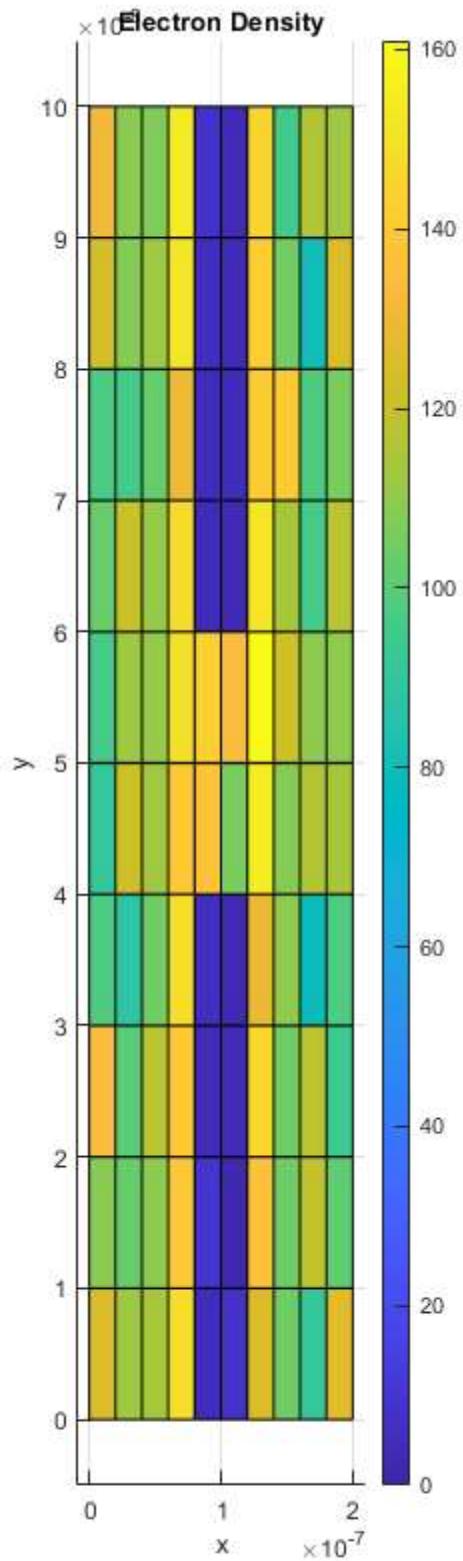
```
end
```

The electron spectral density is created through use of a histogram.

```
figure(3)
hist3([x, y], 'CDataMode', 'auto', 'FaceColor', 'interp');
title('Electron Density')
xlabel('x')
ylabel('y')
colorbar
view(2)
```

```
function traj(electrons, xp, x, yp, y, TrajColours, w, h)
```

```
rectangle('position', [80e-9 60e-9 40e-9 40e-9])
hold on
rectangle('position', [80e-9 0e-9 40e-9 40e-9])
hold on
for k =1:length(electrons)
    x1 = xp(k);
    x2 = x(k);
    y1 = yp(k);
    y2 = y(k);
    colour = TrajColours;
    plot([x1 x2], [y1 y2], 'color', colour(k,:));
    axis([0 w 0 h])
    xlabel('x')
    ylabel('y')
    title(['Electron Simulation']);
end
```



2. Discussion

The thermal velocity was found to be $1.870135757482418e+05$ m/s. The mean free path was found to be $3.740271514964836e-08$ m. From Figure 2 it is evident that over time the average temperature regresses toward 300K the given temperature. A solution

was not found to calculate the mean time between collisions, the mean free path, and a temperature map.

3. Conclusion

In conclusion, the assignment was mostly completed. A simulation of the behaviour of electrons under many different circumstances was designed.

Published with MATLAB® R2020a