Temperature Compensation Report

03/06/2024

- Process temperature compensation for sensors 000006 and 000007 from London plant.
- Other sensors were in condensate.
- Discussion in meeting about sending parameters from the temperature compensation fit.
- Fit a 1^{st} order polynomial (straight-line) to day 1 data and apply to day 2 data.

Imports

- Import Python libraries.

In [ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from pathlib import Path
from datetime import datetime

root = Path('K://Josh/Argus')
```

Data Processing

Split into sensor 000006 and 000007, start with 6.

Reader 6

In [ ]:
```
target_path = Path(f'{root}/Site_Data/London/Reader_000006')
day1 = Path(f'{target_path}/GMR_results_000006_20240601_000012.csv')
day2 = Path(f'{target_path}/GMR_results_000006_20240602_000021.csv')
```

Use pandas library to read CSV with "callable" header columns.

In [ ]:
```
day1_data = pd.read_csv(
    filepath_or_buffer=day1,
    sep=',',
    skiprows=3)
day2_data = pd.read_csv(
    filepath_or_buffer=day2,
    sep=',',
    skiprows=3)
```

Convert time/date stamps into time stamps for both days, this will create sequential data collection.

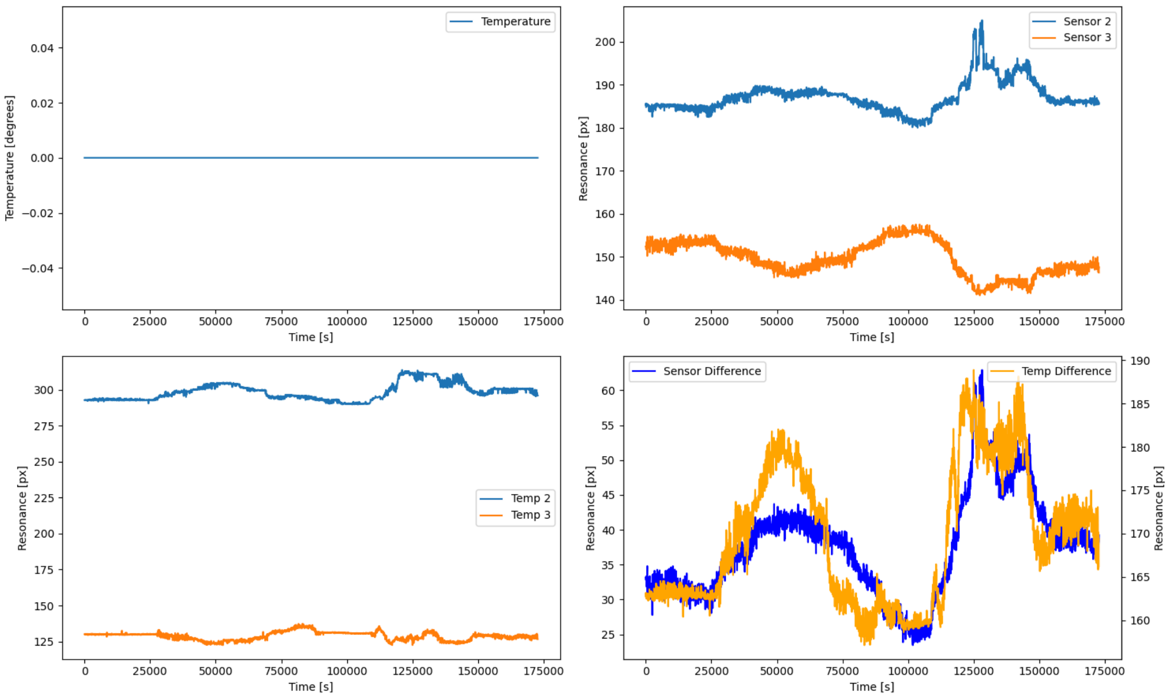
In [ ]:
```
timedate_stamps1 = day1_data['# Date']
time_obj1 = [datetime.strptime(date_str, "%Y%m%d_%H%M%S") for date_str in timeda
timestamps1 = [obj.timestamp() for obj in time_obj1]
corrected_time1 = [time - timestamps1[0] for time in timestamps1]

timedate_stamps2 = day2_data['# Date']
time_obj2 = [datetime.strptime(date_str, "%Y%m%d_%H%M%S") for date_str in timeda
timestamps2 = [obj.timestamp() for obj in time_obj2]
corrected_time2 = [time - timestamps2[0] + corrected_time1[-1] for time in times
```

Concatenate data from both days into full data set.

In [ ]:
```
total_time = np.concatenate((corrected_time1, corrected_time2))
temperature = np.concatenate((day1_data['temperature'], day2_data['temperature']
sensor2 = np.concatenate((day1_data['sensor2'], day2_data['sensor2']))
sensor3 = np.concatenate((day1_data['sensor3'], day2_data['sensor3']))
temp2 = np.concatenate((day1_data['temp2'], day2_data['temp2']))
temp3 = np.concatenate((day1_data['temp3'], day2_data['temp3']))
```

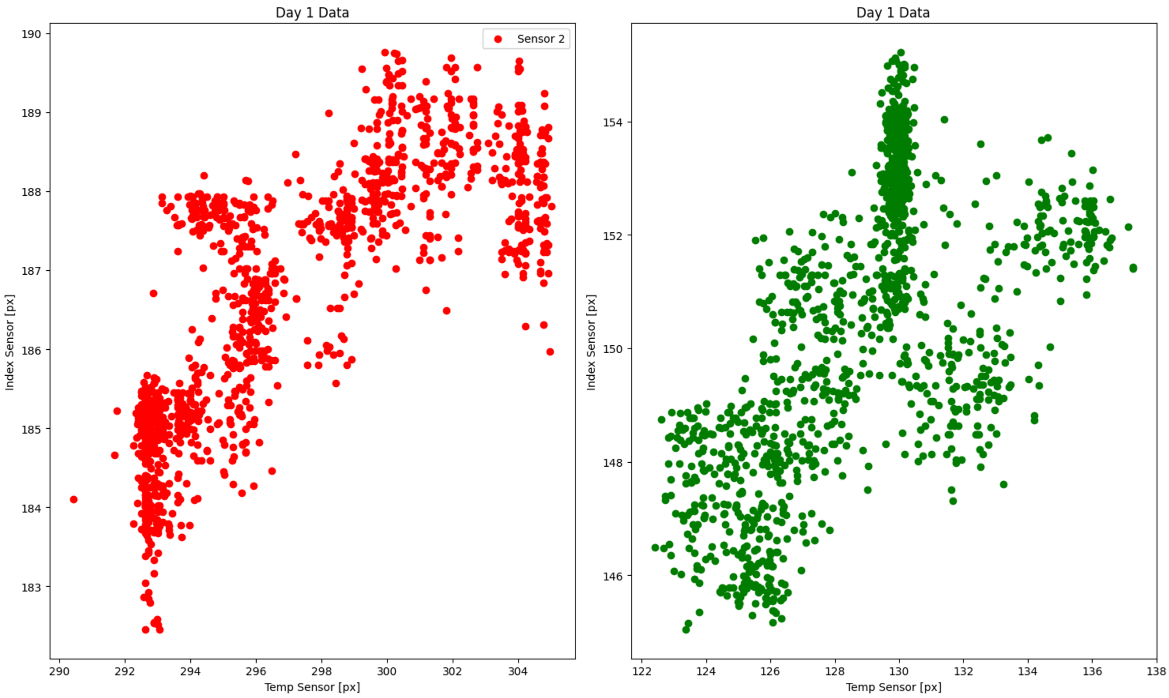
Plot raw data to show good inputting of the parameter space.

In [ ]:
```
fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(
    nrows=2,
    ncols=2,
    figsize=[15, 9])
ax1.plot(total_time, temperature, label='Temperature')
ax1.legend(loc=0)
ax1.set_xlabel('Time [s]')
ax1.set_ylabel('Temperature [degrees]')
ax2.plot(total_time, sensor2, label='Sensor 2')
ax2.plot(total_time, sensor3, label='Sensor 3')
ax2.legend(loc=0)
ax2.set_xlabel('Time [s]')
ax2.set_ylabel('Resonance [px]')
ax3.plot(total_time, temp2, label='Temp 2')
ax3.plot(total_time, temp3, label='Temp 3')
ax3.legend(loc=0)
ax3.set_xlabel('Time [s]')
ax3.set_ylabel('Resonance [px]')
ax5 = ax4.twinx()
ax4.plot(total_time, sensor2-sensor3, 'b', label='Sensor Difference')
ax5.plot(total_time, temp2-temp3, 'orange', label='Temp Difference')
ax4.legend(loc=0)
ax5.legend(loc=1)
ax4.set_xlabel('Time [s]')
ax4.set_ylabel('Resonance [px]')
ax5.set_ylabel('Resonance [px]')
fig.tight_layout()
plt.show()
```



Assume that the sensors are being outputted from the bow tie software, hence sensor 2 and sensor 3 (which are the only ones with data associated with them) are the result of a bow tie measurement.

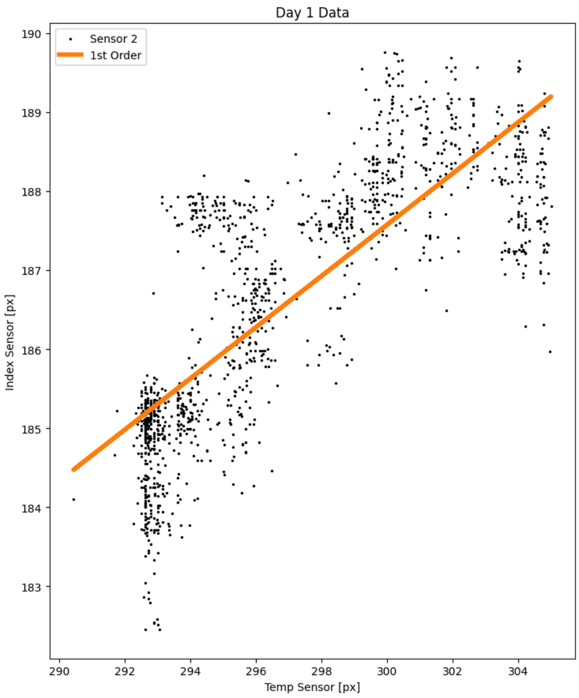
In [ ]:
```
fig, (ax1, ax2) = plt.subplots(nrows=1, ncols=2, figsize=[15, 9])
ax1.scatter(day1_data['temp2'], day1_data['sensor2'], color='r', label='Sensor 2
ax2.scatter(day1_data['temp3'], day1_data['sensor3'], color='g', label='Sensor 3
ax1.set_xlabel('Temp Sensor [px]')
ax1.set_ylabel('Index Sensor [px]')
ax2.set_xlabel('Temp Sensor [px]')
ax2.set_ylabel('Index Sensor [px]')
ax1.legend(loc=0)
ax1.set_title('Day 1 Data')
ax2.set_title('Day 1 Data')
fig.tight_layout()
plt.show()
```

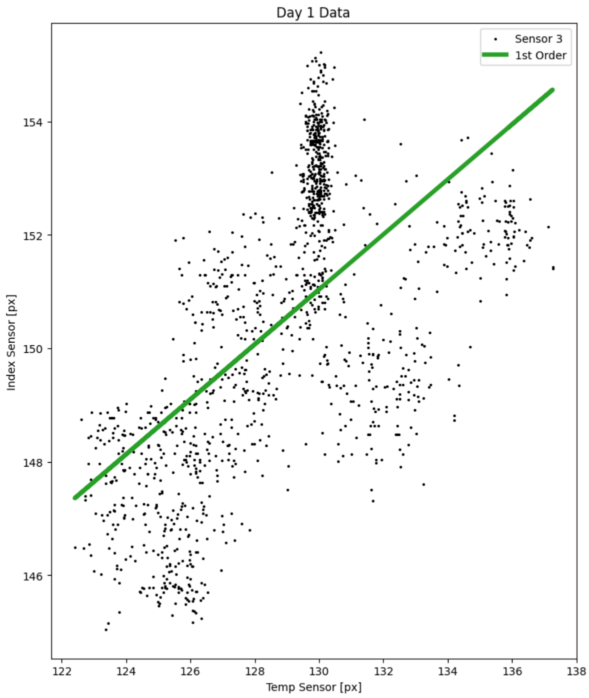


Fit a straight line to each of the sensor data to get the calibration parameters.

```
In [ ]: degree = 1

        x_2 = day1_data['temp2']
        y_2 = day1_data['sensor2']

        x_3 = day1_data['temp3']
        y_3 = day1_data['sensor3']

        coefficients_2 = np.polyfit(x_2, y_2, degree)
        coefficients_3 = np.polyfit(x_3, y_3, degree)

        polynomial_2 = np.poly1d(coefficients_2)
        polynomial_3 = np.poly1d(coefficients_3)

        print(polynomial_2, polynomial_3)

        fig, (ax1, ax2) = plt.subplots(nrows=1, ncols=2, figsize=[15, 9])
        ax1.scatter(day1_data['temp2'], day1_data['sensor2'], color='k', s=2, label='Sen
        ax2.scatter(day1_data['temp3'], day1_data['sensor3'], color='k', s=2, label='Sen
        ax1.set_xlabel('Temp Sensor [px]')
        ax1.set_ylabel('Index Sensor [px]')
        ax2.set_xlabel('Temp Sensor [px]')
        ax2.set_ylabel('Index Sensor [px]')
        ax1.plot(x_2, polynomial_2(x_2), f'C1', lw=4, label=f'1st Order')
        ax2.plot(x_3, polynomial_3(x_3), f'C2', lw=4, label=f'1st Order')
        ax1.legend(loc=0)
        ax2.legend(loc=0)
        ax1.set_title('Day 1 Data')
        ax2.set_title('Day 1 Data')
        fig.tight_layout()
        plt.show()
```

```
0.3243 x + 90.27
0.4848 x + 88.01
```



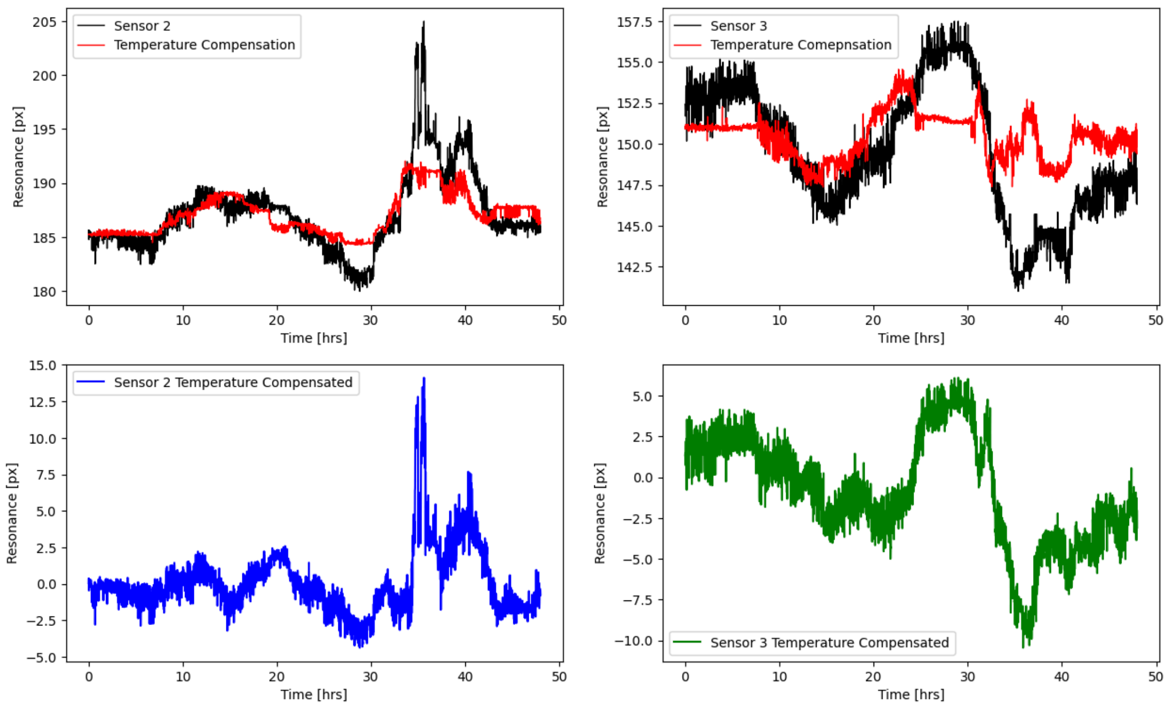


Here, the temperature compensation parameters for sensor 2 and 3 on reader 6 are:

Sensor	m	c
2	0.3243	90.27
3	0.4848	88.01

Now use these to correct the data on day 2.

In [ ]:
```
compensated_sensor2 = sensor2 - polynomial_2(temp2)
compensated_sensor3 = sensor3 - polynomial_3(temp3)

total_hours = total_time / (60 * 60)

fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(nrows=2, ncols=2, figsize=[15, 9])
ax1.plot(total_hours, sensor2, 'k', lw=1, label='Sensor 2')
ax2.plot(total_hours, sensor3, 'k', lw=1, label='Sensor 3')
ax3.plot(total_hours, compensated_sensor2, 'b', label='Sensor 2 Temperature Comp
ax4.plot(total_hours, compensated_sensor3, 'g', label='Sensor 3 Temperature Comp
ax1.plot(total_hours, polynomial_2(temp2), 'r', lw=1, label='Temperature Compens
ax2.plot(total_hours, polynomial_3(temp3), 'r', lw=1, label='Temperature Comepns
ax1.legend(loc=0)
ax2.legend(loc=0)
ax3.legend(loc=0)
ax4.legend(loc=0)
ax1.set_xlabel('Time [hrs]')
ax1.set_ylabel('Resonance [px]')
ax2.set_xlabel('Time [hrs]')
ax2.set_ylabel('Resonance [px]')
ax3.set_xlabel('Time [hrs]')
ax3.set_ylabel('Resonance [px]')
ax4.set_xlabel('Time [hrs]')
ax4.set_ylabel('Resonance [px]')
plt.show()
```



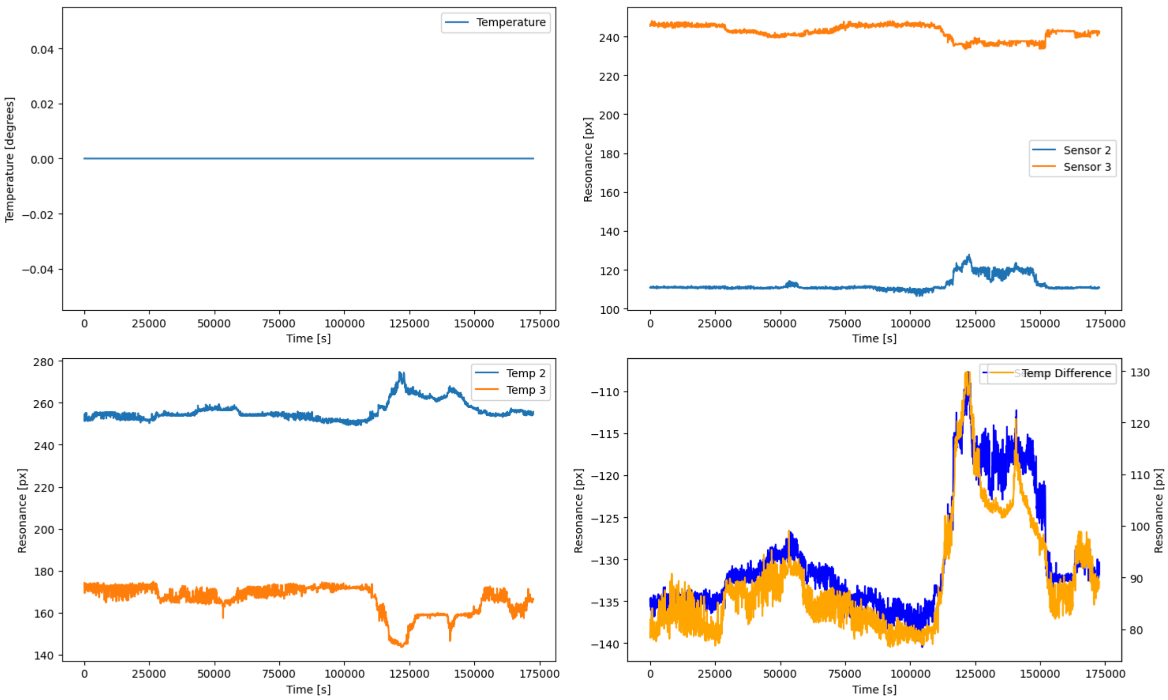
Repeat process for reader 000007.

Reader 7

In [ ]:
```
target_path = Path(f'{root}/Site_Data/London/Reader_000007')
day1 = Path(f'{target_path}/GMR_results_000007_20240601_000036.csv')
day2 = Path(f'{target_path}/GMR_results_000007_20240602_000023.csv')

day1_data = pd.read_csv(
    filepath_or_buffer=day1,
    sep=',',
    skiprows=3)
day2_data = pd.read_csv(
    filepath_or_buffer=day2,
    sep=',',
    skiprows=3)

timedate_stamps1 = day1_data['# Date']
time_obj1 = [datetime.strptime(date_str, "%Y%m%d_%H%M%S") for date_str in timeda
timestamps1 = [obj.timestamp() for obj in time_obj1]
corrected_time1 = [time - timestamps1[0] for time in timestamps1]
timedate_stamps2 = day2_data['# Date']
time_obj2 = [datetime.strptime(date_str, "%Y%m%d_%H%M%S") for date_str in timeda
timestamps2 = [obj.timestamp() for obj in time_obj2]
corrected_time2 = [time - timestamps2[0] + corrected_time1[-1] for time in times

total_time = np.concatenate((corrected_time1, corrected_time2))
temperature = np.concatenate((day1_data['temperature'], day2_data['temperature']
sensor2 = np.concatenate((day1_data['sensor2'], day2_data['sensor2']))
sensor3 = np.concatenate((day1_data['sensor3'], day2_data['sensor3']))
temp2 = np.concatenate((day1_data['temp2'], day2_data['temp2']))
temp3 = np.concatenate((day1_data['temp3'], day2_data['temp3']))
```

Process the raw data in the same way.

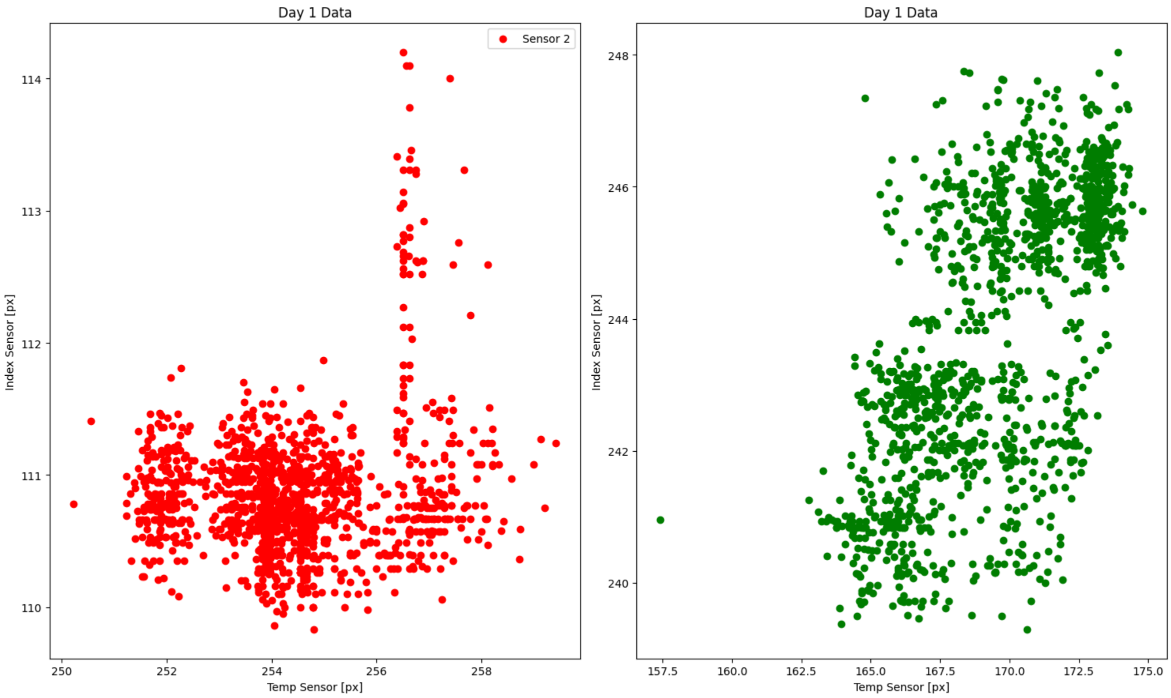
In [ ]:
```
fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(
    nrows=2,
    ncols=2,
    figsize=[15, 9])
ax1.plot(total_time, temperature, label='Temperature')
ax1.legend(loc=0)
ax1.set_xlabel('Time [s]')
ax1.set_ylabel('Temperature [degrees]')
ax2.plot(total_time, sensor2, label='Sensor 2')
ax2.plot(total_time, sensor3, label='Sensor 3')
ax2.legend(loc=0)
ax2.set_xlabel('Time [s]')
ax2.set_ylabel('Resonance [px]')
ax3.plot(total_time, temp2, label='Temp 2')
ax3.plot(total_time, temp3, label='Temp 3')
ax3.legend(loc=0)
ax3.set_xlabel('Time [s]')
ax3.set_ylabel('Resonance [px]')
ax5 = ax4.twinx()
ax4.plot(total_time, sensor2-sensor3, 'b', label='Sensor Difference')
ax5.plot(total_time, temp2-temp3, 'orange', label='Temp Difference')
ax4.legend(loc=0)
ax5.legend(loc=1)
ax4.set_xlabel('Time [s]')
ax4.set_ylabel('Resonance [px]')
ax5.set_ylabel('Resonance [px]')
fig.tight_layout()
plt.show()
```



Now the temperature compensation.

In [ ]:
```
fig, (ax1, ax2) = plt.subplots(nrows=1, ncols=2, figsize=[15, 9])
ax1.scatter(day1_data['temp2'], day1_data['sensor2'], color='r', label='Sensor 2
ax2.scatter(day1_data['temp3'], day1_data['sensor3'], color='g', label='Sensor 3
ax1.set_xlabel('Temp Sensor [px]')
ax1.set_ylabel('Index Sensor [px]')
ax2.set_xlabel('Temp Sensor [px]')
```

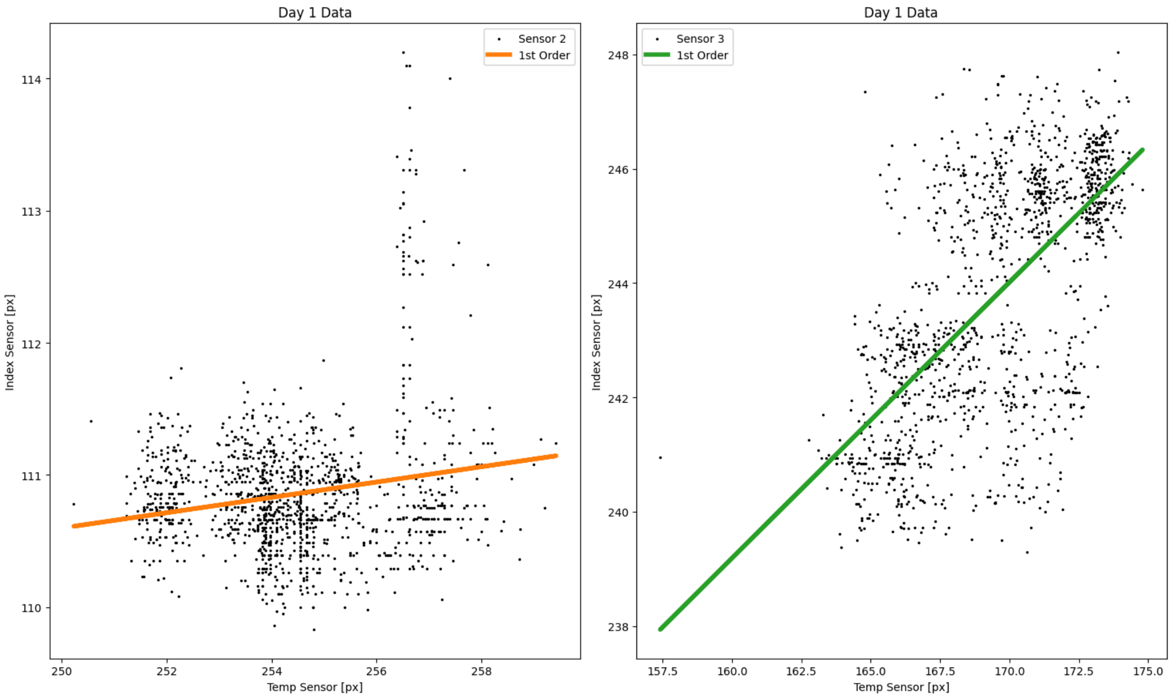
```
ax2.set_ylabel('Index Sensor [px]')
ax1.legend(loc=0)
ax1.set_title('Day 1 Data')
ax2.set_title('Day 1 Data')
fig.tight_layout()
plt.show()
```



Fit the straight-line.

In [ ]:
```
degree = 1
x_2 = day1_data['temp2']
y_2 = day1_data['sensor2']
x_3 = day1_data['temp3']
y_3 = day1_data['sensor3']
coefficients_2 = np.polyfit(x_2, y_2, degree)
coefficients_3 = np.polyfit(x_3, y_3, degree)
polynomial_2 = np.poly1d(coefficients_2)
polynomial_3 = np.poly1d(coefficients_3)
print(polynomial_2, polynomial_3)

fig, (ax1, ax2) = plt.subplots(nrows=1, ncols=2, figsize=[15, 9])
ax1.scatter(day1_data['temp2'], day1_data['sensor2'], color='k', s=2, label='Sen
ax2.scatter(day1_data['temp3'], day1_data['sensor3'], color='k', s=2, label='Sen
ax1.set_xlabel('Temp Sensor [px]')
ax1.set_ylabel('Index Sensor [px]')
ax2.set_xlabel('Temp Sensor [px]')
ax2.set_ylabel('Index Sensor [px]')
ax1.plot(x_2, polynomial_2(x_2), f'C1', lw=4, label=f'1st Order')
ax2.plot(x_3, polynomial_3(x_3), f'C2', lw=4, label=f'1st Order')
ax1.legend(loc=0)
ax2.legend(loc=0)
ax1.set_title('Day 1 Data')
ax2.set_title('Day 1 Data')
fig.tight_layout()
plt.show()
```

```
0.05802 x + 96.09
0.4828 x + 161.9
```



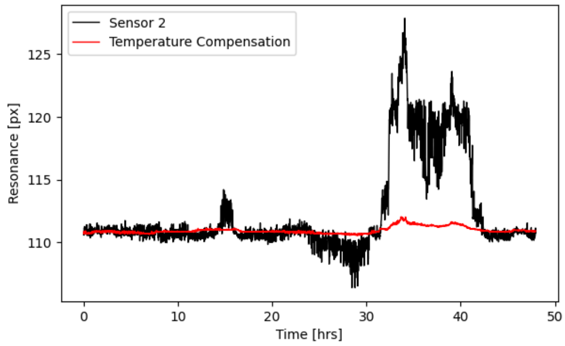
Here, the temperature compensation parameters for sensor 2 and 3 on reader 6 are:

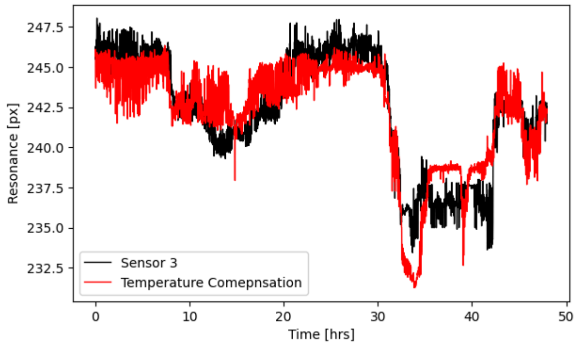
Sensor	m	c
2	0.05802	96.09
3	0.4828	161.9

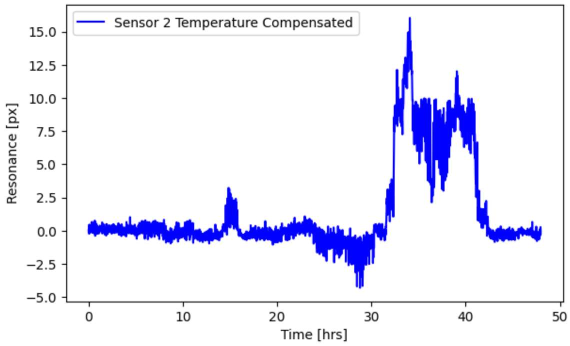
Now use these to correct the data on day 2.

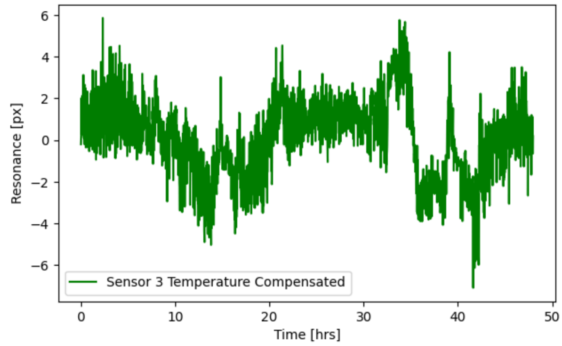
In [ ]:

```
compensated_sensor2 = sensor2 - polynomial_2(temp2)
compensated_sensor3 = sensor3 - polynomial_3(temp3)
total_hours = total_time / (60 * 60)
fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(nrows=2, ncols=2, figsize=[15, 9])
ax1.plot(total_hours, sensor2, 'k', lw=1, label='Sensor 2')
ax2.plot(total_hours, sensor3, 'k', lw=1, label='Sensor 3')
ax3.plot(total_hours, compensated_sensor2, 'b', label='Sensor 2 Temperature Comp
ax4.plot(total_hours, compensated_sensor3, 'g', label='Sensor 3 Temperature Comp
ax1.plot(total_hours, polynomial_2(temp2), 'r', lw=1, label='Temperature Compens
ax2.plot(total_hours, polynomial_3(temp3), 'r', lw=1, label='Temperature Comepns
ax1.legend(loc=0)
ax2.legend(loc=0)
ax3.legend(loc=0)
ax4.legend(loc=0)
ax1.set_xlabel('Time [hrs]')
ax1.set_ylabel('Resonance [px]')
ax2.set_xlabel('Time [hrs]')
ax2.set_ylabel('Resonance [px]')
ax3.set_xlabel('Time [hrs]')
ax3.set_ylabel('Resonance [px]')
ax4.set_xlabel('Time [hrs]')
ax4.set_ylabel('Resonance [px]')
plt.show()
```


Sensor 2
Temperature Compensation
Resonance [px]
Time [hrs]
110
115
120
125
0
10
20
30
40
50


Sensor 3
Temperature Comepnsation
Resonance [px]
Time [hrs]
232.5
235.0
237.5
240.0
242.5
245.0
247.5
0
10
20
30
40
50


Sensor 2 Temperature Compensated
Resonance [px]
Time [hrs]
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
12.5
15.0
0
10
20
30
40
50


Sensor 3 Temperature Compensated
Resonance [px]
Time [hrs]
−6
−4
−2
0
2
4
6
0
10
20
30
40
50