Particle Resuspension Detection

Michael Gachich, Nicholas Jung, Samriddhi Bhatia, Jake Rosenbaum

Background

- In the past 45 years there have been three major events (such as Chernobyl) that spread radioactive particles throughout the nearby landscape. These particles eventually settle on the ground
- However, vehicles and people entering these contaminated areas carry the risk of resuspending these particles in the air, causing radiation sickness and death at elevated rates

https://www.livescience.com/39961-chernobyl.html

https://telanganatoday.com/here-are-some-shocking-facts-about-fukushima-nuclear-tragedy

Purpose

- To examine the characteristics and behavior of particle resuspension in an urban environment

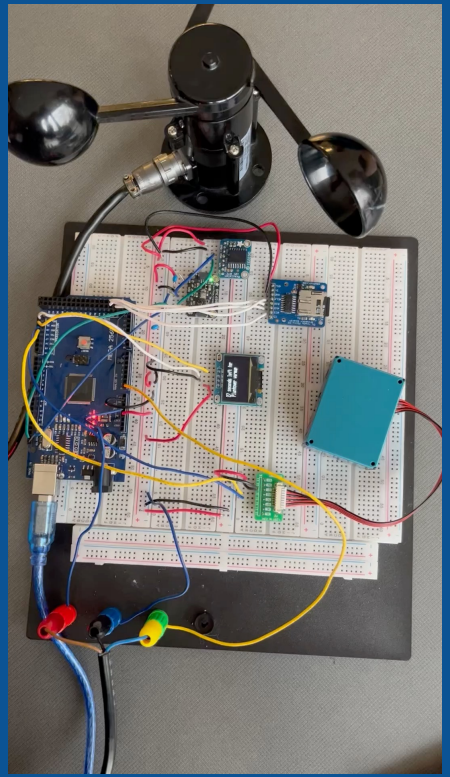
- To replicate the functionality and efficiency of commercially available air quality monitors.

- To develop a small, portable, and low-power device to monitor the counts of dust particles, temperature, humidity, and pressure.

- To integrate an anemometer to measure wind speed.

Sensors/Components - What We Are Measuring

- Plantower: Particulate matter sensor, bins of >1, >2.5, >5, >10 um.
- BME680: Pressure (hPa), temperature (C), humidity (%).
- Cup Anemometer: Wind speed (in m/s and converted to mph).
- RTC: Time monitoring of resuspension using data timestamps.
- OLED: Display status messages about the sensors and collection process.



Schematic and Housing Considerations

- Plantower: To be kept at a variable orientation, UART protocol

- BME680: Adequate atmospheric exposure,

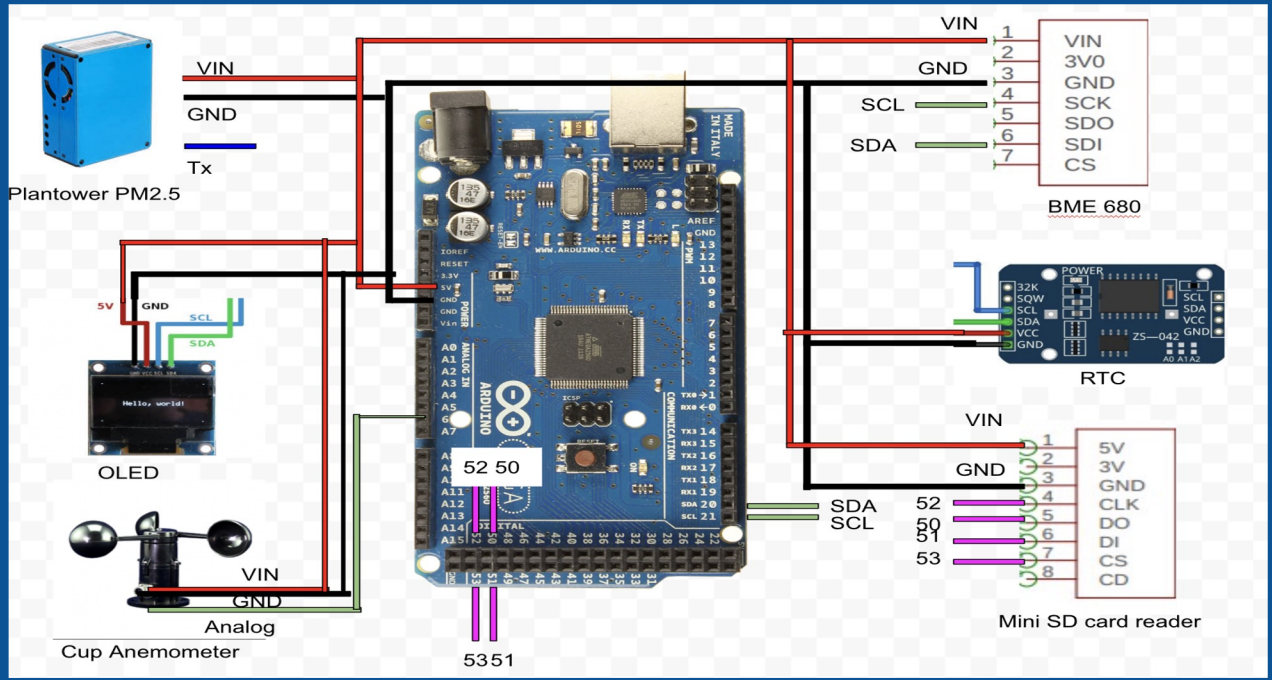
  I2C communication

- Cup Anemometer:

  External mounting,

  analog voltage output

- RTC: I2C communication.



Code

What it does:

Arduino:

- Initialize sensors
- Checks if sensors work
- Displays status on OLED
- Writes data from BME, Plantower, etc to SD

Python:

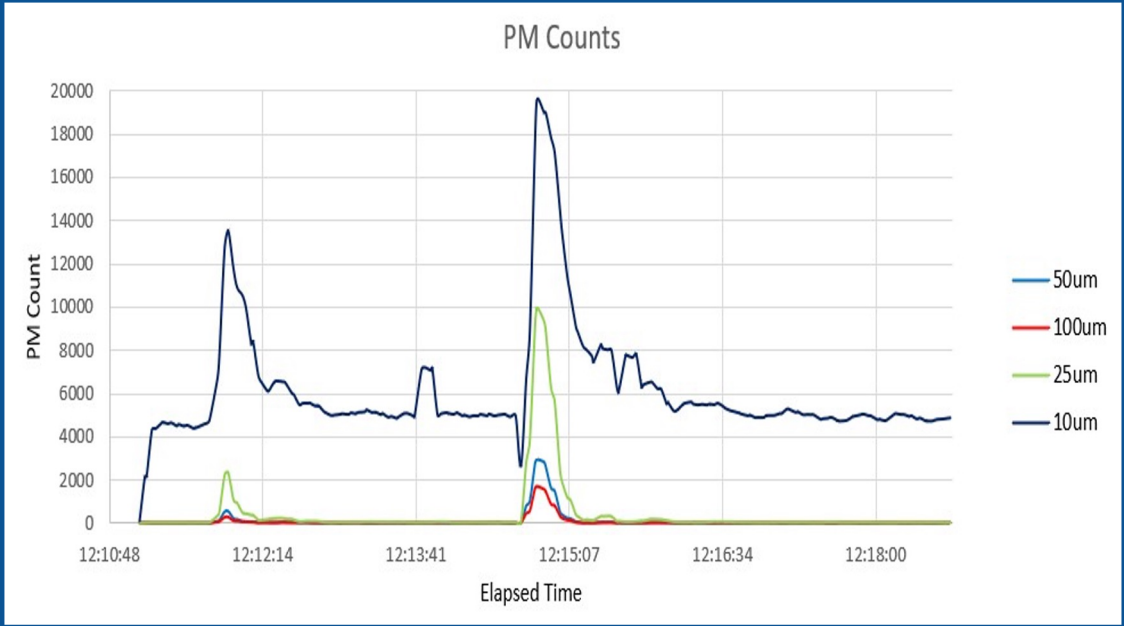
- Reads CSV file from SD
- Parses data into a Dictionary
- Create plots/further data analysis

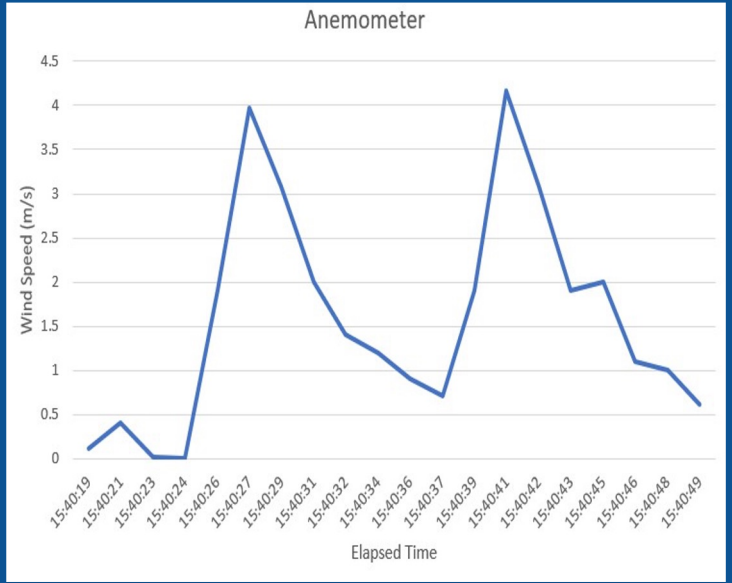
```python
import numpy as np
import matplotlib.pyplot as plt
from pandas import *
# Parse the data
data = read_csv("DAQ.csv", usecols = [0,1,2,3,4,5,6], names = ['Temp', 'Humidity', 'Pressure', 'Pm10', 'Pm25','Pm100', 'Part_03um'])

temp = data['Temp'].to_numpy()
hum =  data['Humidity'].to_numpy()
pressure = data['Pressure'].to_numpy()
pm10 =  data['Pm10'].to_numpy()
pm25 =  data['Pm25'].to_numpy()
pm100 =  data['Pm100'].to_numpy()
p03_um = data['Part_03um'].to_numpy()
p05_um = data['Part_05um'].to_numpy()
p10_um = data['Part-10um'].to_numpy()

# Will Plot Temperature as that changes somewhat
x = temp
y = np.arange(len(x))
avg = np.average(temp)
np.format_float_positional(avg, precision = 2)
plt.figure(dpi = 800)
plt.plot(y,x)
plt.text(15, 27, "Average = " + str(avg) + " C")
plt.xlabel("Time (seconds)")
plt.ylabel("Temperature (C)")
plt.title("Temperature fluctuations in my apartment")

x1 = pm10
x2 = pm25
x3 = pm100
plt.figure(dpi = 1200)
plt.plot(pm25)
```

Data Collection and Management

- Plantower calibration done in controlled, minimal PM environment.
- Outside test runs done with mechanically disturbed particulates, data collected over extended time intervals ~15 minutes.
- Data collection performed at roughly 1 Hz, 900 data points.
- Recorded data locally onto SD card into a .csv file
- Processed data in Python for data analysis/plotting

Data Analysis





Test data for Plantower (using a soldering iron) and cup anemometer under controlled conditions.

Next Steps

- Wireless connectivity for real-time data reading
- PCB design + housing design
  - Weatherproofing
  - Variable Plantower positioning
  - Access to SD + a button to cycle through OLED
- External battery / power source