GNG 5140

Design Project User and Product Manual

Personal Tracking System

Submitted by:

Personal Tracking System, Group B

Mohammad Reza Hajirezaei 300322306

Mohammad Waleed Qanbar 300322743

Dalileh Mehrabi 300261570

Daniyal Sojoudi 300289816

April 25, 2023

University of Ottawa

Table of Contents

List of Figures

List of Tables

List of Acronyms and Glossary

Table 1. Acronyms

Acronym	Definition
AQMS	Air Quality Monitoring System
GPS	Global Positioning System
PM	Particulate Matter
UPM	User and Product Manual

1 Introduction

Air pollution is a significant environmental issue affecting cities worldwide. The World Health Organization (WHO) estimates that air pollution is responsible for 7 million deaths annually, and 90% of people breathe polluted air. In response to this issue, governments, organizations, and individuals are taking action to monitor air quality, identify sources of pollution, and mitigate their impacts.

The AQMS is a prototype designed to measure air quality in Ottawa, Canada. It uses PMS5003 and MQ135 sensors, GPS (Global Positioning System), and Wi-Fi modules to collect data on air pollutants, including PM2.5 and carbon dioxide. A microcontroller ESP32 board which has built-in Wi-Fi and Bluetooth capabilities is used for connecting components. The system is powered by a laptop and the collected data is visualized using Python and Power BI. The AQMS prototype is an excellent tool for monitoring air pollution and raising awareness about its effects.

This UPM provides comprehensive information necessary for diverse types of users to effectively use and maintain the AQMS prototype and collect and visualize air pollution data. The document covers the technical aspects of the prototype, including its functionality, limitations, and safety considerations, and is intended for researchers, students, and organizations interested in air quality monitoring. It assumes that users have basic knowledge of electronics, programming, and data visualization. Also, security and privacy considerations associated with the use of the AQMS are addressed in this manual to ensure safe and secure handling of data. The prototype is designed to be low-cost, easy to use, and customizable to meet the needs of various users.

This manual serves as a guide for the proper use and maintenance of the prototype and for collecting air pollution data and visualizing them. It provides a detailed overview of the AQMS, including its components and hardware requirements, including the sensors, microcontroller, and GPS device and their connections and how to collect data. As well as focus on data visualization and analysis, detailing how to store and analyze the collected data to create pollution maps and identify trends in air quality. We will provide instructions on how to use both Power BI and Python to visualize the collected data. The manual also includes information on maintaining and troubleshooting the

prototype and safety considerations when working with electronic components and handling data. This manual explains how to assemble the system and test its functionality.

By using this manual, users can continue this project in various aspects. Some further steps in this project include uploading the collected data to the cloud and visualizing and analyzing them in real time. Using the cloud lets users synchronize several AQMS and collect more data. This manual provides a starting point for users interested in customizing and expanding the prototype to meet their specific needs.

2 Overview

A significant environmental issue that has an impact on both human and environmental health is air pollution. Cardiovascular illness, respiratory issues, and early death are all related to poor air quality. Additionally, air pollution harms ecosystems and accelerates climate change. In order to comprehend the effects that air pollution has on our environment and health, it is crucial to precisely monitor and measure it.

For this project, the user's fundamental needs are an air pollution monitoring tool that is simple to use, accurate, and portable so it may be carried while cycling or on a mobile device. The instrument should include real-time hydrocarbon and PM2.5 level measurement capabilities, as well as GPS and Wi-Fi connectivity for sending data to a central database for analysis and visualization.

The key aspects of this project's prototype are its portability and capacity to gather real-time information on air pollution. The use of a microcontroller and sensors provides an affordable and accessible solution to monitoring air pollution, making it ideal for individuals and organizations that want to collect data on air quality. Making decisions can also be made with the use of data visualization.

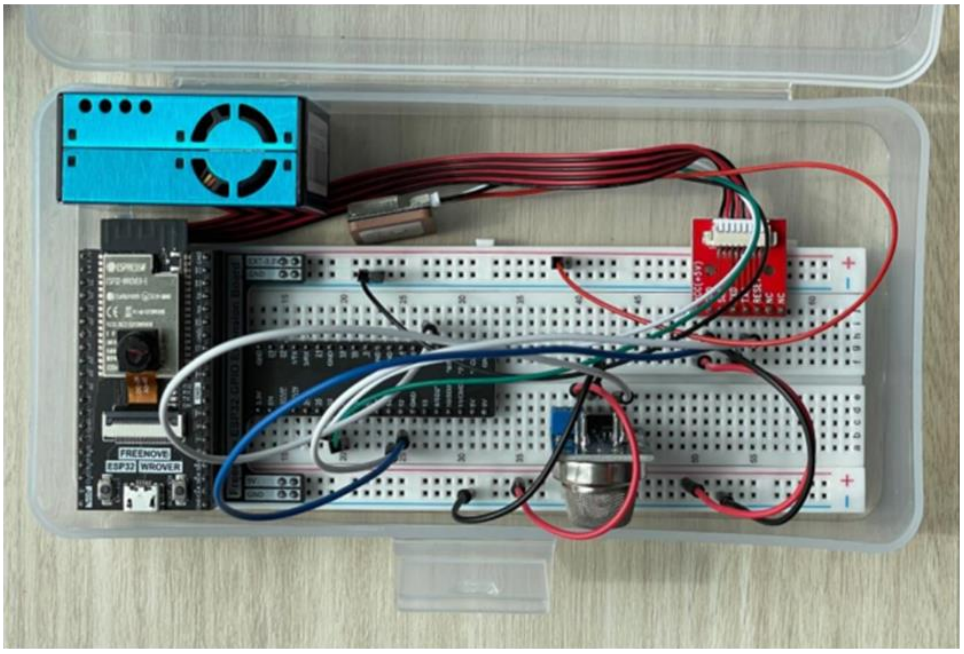

Figure 1: Final Prototype Picture

Real-time measurements of PM 2.5 and hydrocarbon levels, GPS and Wi-Fi connectivity, and data visualization capabilities are among the product's primary features. Anyone may use the product easily thanks to its user-friendly interface.

The architecture of the system includes an ESP32 board that acts as a microcontroller for collecting sensor data. Using Wi-Fi connectivity, the data is then transmitted to a central database where it is analyzed and visualized. In the next step of this project, users will be able to access the data via a web-based interface on their mobile devices or computers.

Evolving the project in the next step will provide a web-based interface that offers a simple graphical user interface (GUI) for accessing and viewing data used for user access mode. The technology is adaptable and it can also be utilized in an indoor environment based on the target. The system is designed to withstand different weather conditions and can be mounted on a bicycle.

Finally, a block diagram of the system is shown below to provide a visual representation of the different components and how they interact.

Arduino Board → Hydrocarbon Sensor → PM 2.5 Sensor → GPS Module → Wi-Fi Module → Database → Web-based Interface (GUI)

2.1 Cautions & Warnings

- Be careful about the location of the Excel output in your computer and make sure to give the local address of the file in your computer.
- Install the Python, anaconda, and Python libraries.
- You may need to restart the kernel to make the code work.
- Do not expose the AQMS to water or moisture, as this can cause damage to the electronics and render the device inoperable.
- To avoid injury or equipment damage, use caution when handling or working with components, and follow proper safety procedures to prevent injury or damage to the equipment.

- Do not use the AQMS in areas with high temperatures, extreme cold, or other harsh environmental conditions, as this can cause damage or affect the accuracy of the readings.
- Be aware that the AQMS is not water resistant and should not be used in wet or humid environments.
- Before employing sensors to gather data, make sure they are properly calibrated. incorrect calibration can lead to wrong results and inaccurate measurements.
- Attention to environmental factors that could affect data, including wind, dust, temperature, humidity and so on.
- Consistently gather and store data in line with all relevant privacy laws and regulations. Obtain any necessary waivers or permissions before collecting data from individuals.
- Make sure the project supports sustainability and social responsibility and is in line with moral principles.
- Maintaining and calibrating equipment on a regular basis will provide precise and trustworthy results. For future reference, keep a record of the calibration and maintenance processes.
- Privacy or monitoring issues may arise if the system is used in public areas. Should find solutions for these worries.

3 Getting started

3.1 Set-up Considerations

3.1.1 Installing CH340 Driver

Check if the CH340 driver is installed. It is a piece of software that connects the microcontroller chip on the ESP32 to the computer to facilitate communication. Please follow the steps below to check whether it was successfully installed:

1) Connect the ESP32 to your computer via the USB port.

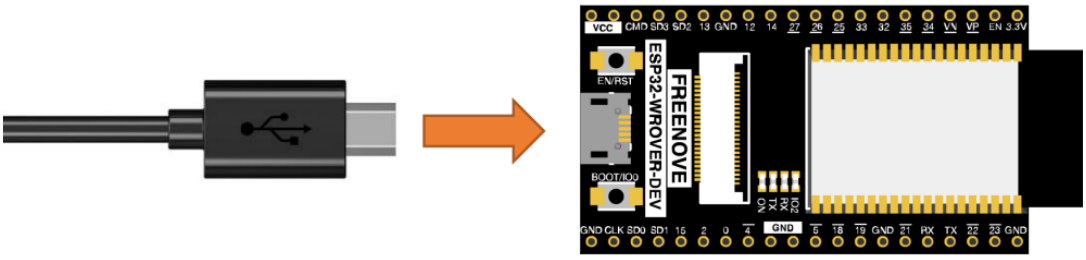

Figure 2: Connecting the ESP to a computer

2) Search for “Device Manager” using the search tool on your Windows Taskbar.

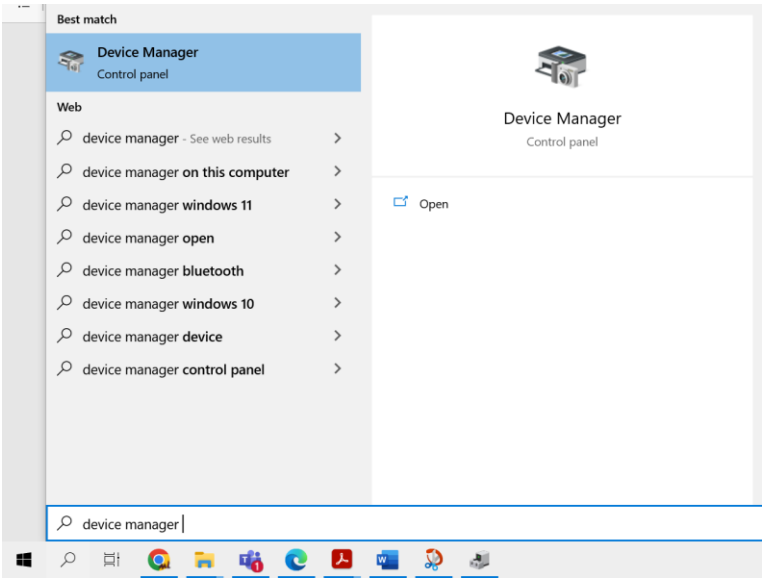

Figure 3: Search the Device Manager

3) Under “Ports (COM & LPT),” check if “USB-SERIAL CH340” is installed.

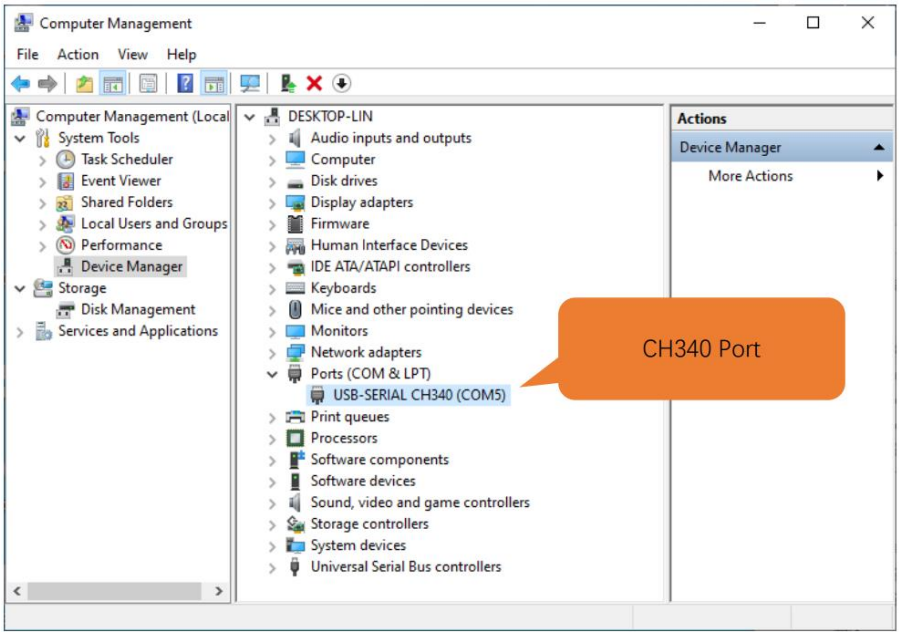

Figure 4: Verify that the CH340 port is installed

4) If not installed, please follow the steps listed on https://sparks.gogo.co.nz/ch340.html.

3.1.2 Installing Arduino IDE

The Arduino IDE is required to upload code to our microcontroller device. To install it, please follow the steps below:

1) Visit https://www.arduino.cc/en/software and download the installer file.

Figure 5: Downloading the Arduino IDE

2) Run the installer and follow the prompts that appear on your screen.

3.1.3 Installing ESP32 Package

After installing Arduino IDE, follow these steps to configure the software for ESP32 coding:

1) On Arduino IDE, click on “File,” then “Preferences.”

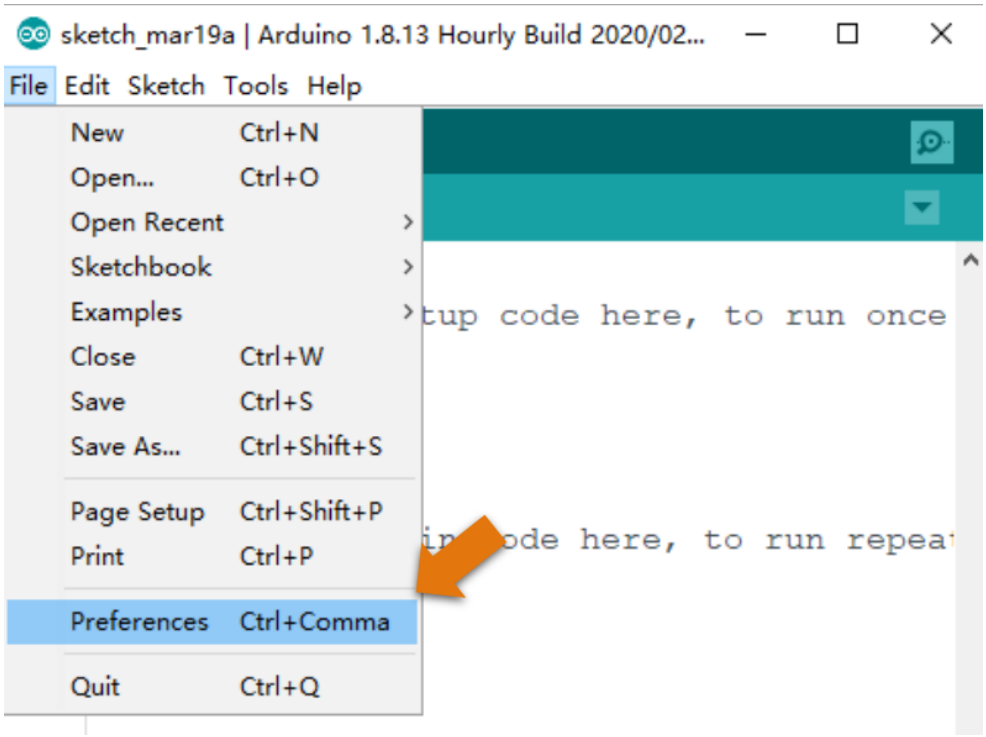

Figure 6: Selecting the preference

2) Click on the symbol to the right of the “Additional Boards Manager URLs” field.

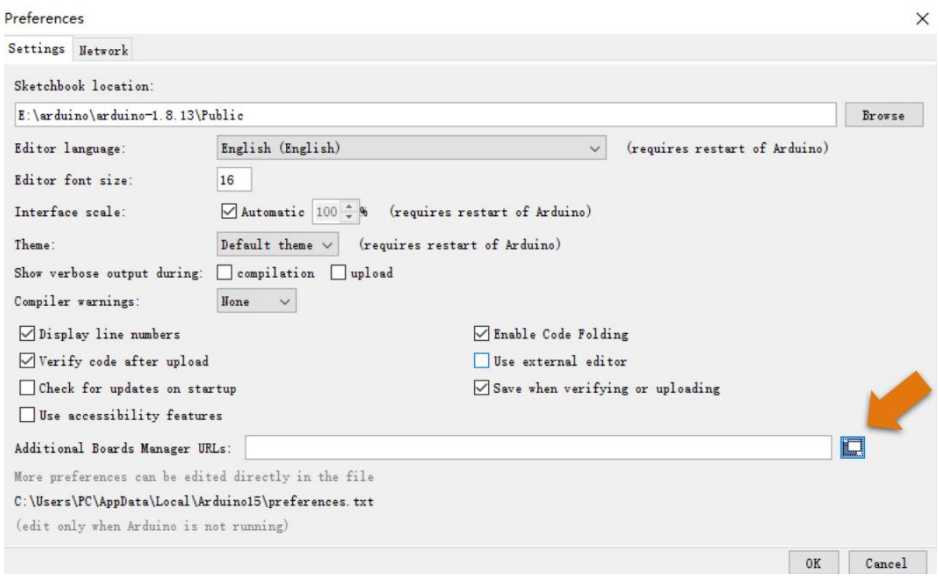

Figure 7: Additional Boards Manager URLs

3) In the new window that pops up, enter the following URL https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json, then click ok.

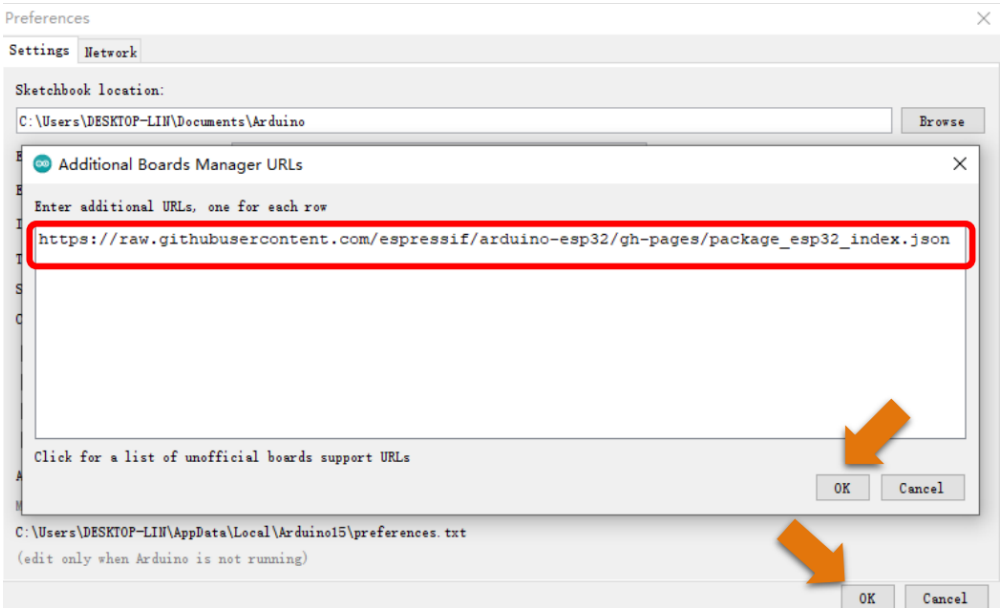

Figure 8: ESP32 Package URL

4) Go back to the main interface and click on “Tools,” then select “Boards Manager.”

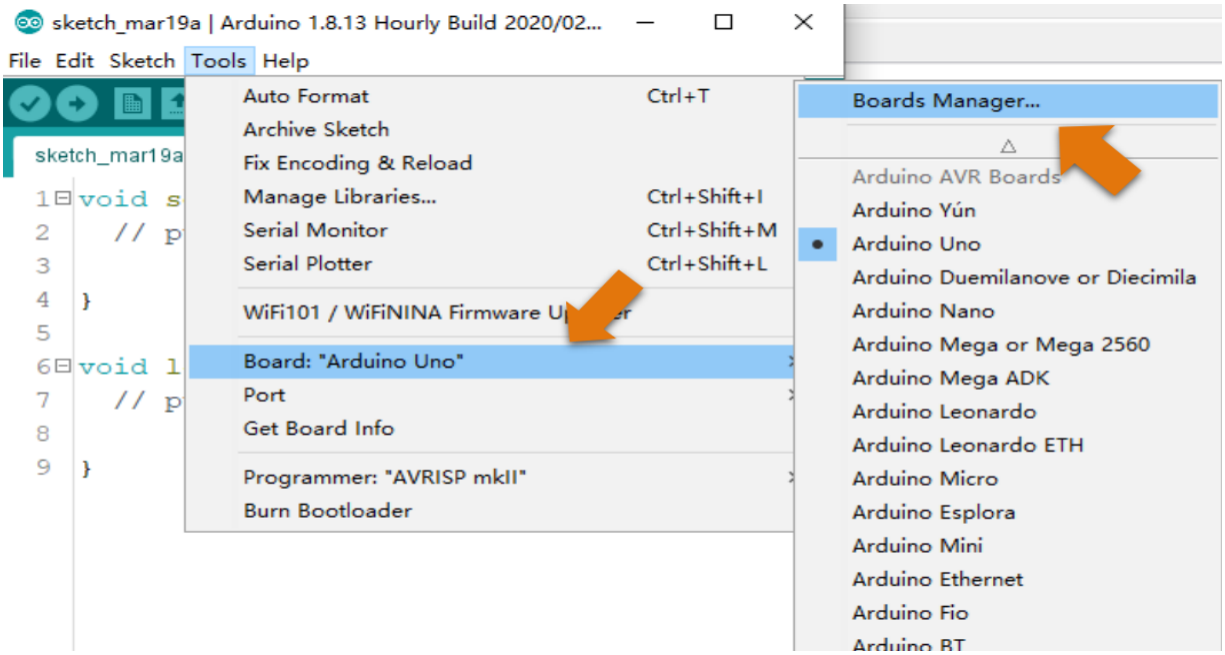

Figure 9: Selecting Board Manager

5) Search for “esp32” in the search bar, then click on “install”.

Figure 10: Installing the EPS32

6) Go back to “Tools,” “Boards” then make sure that “ESP32 Wrover Module” is selected.

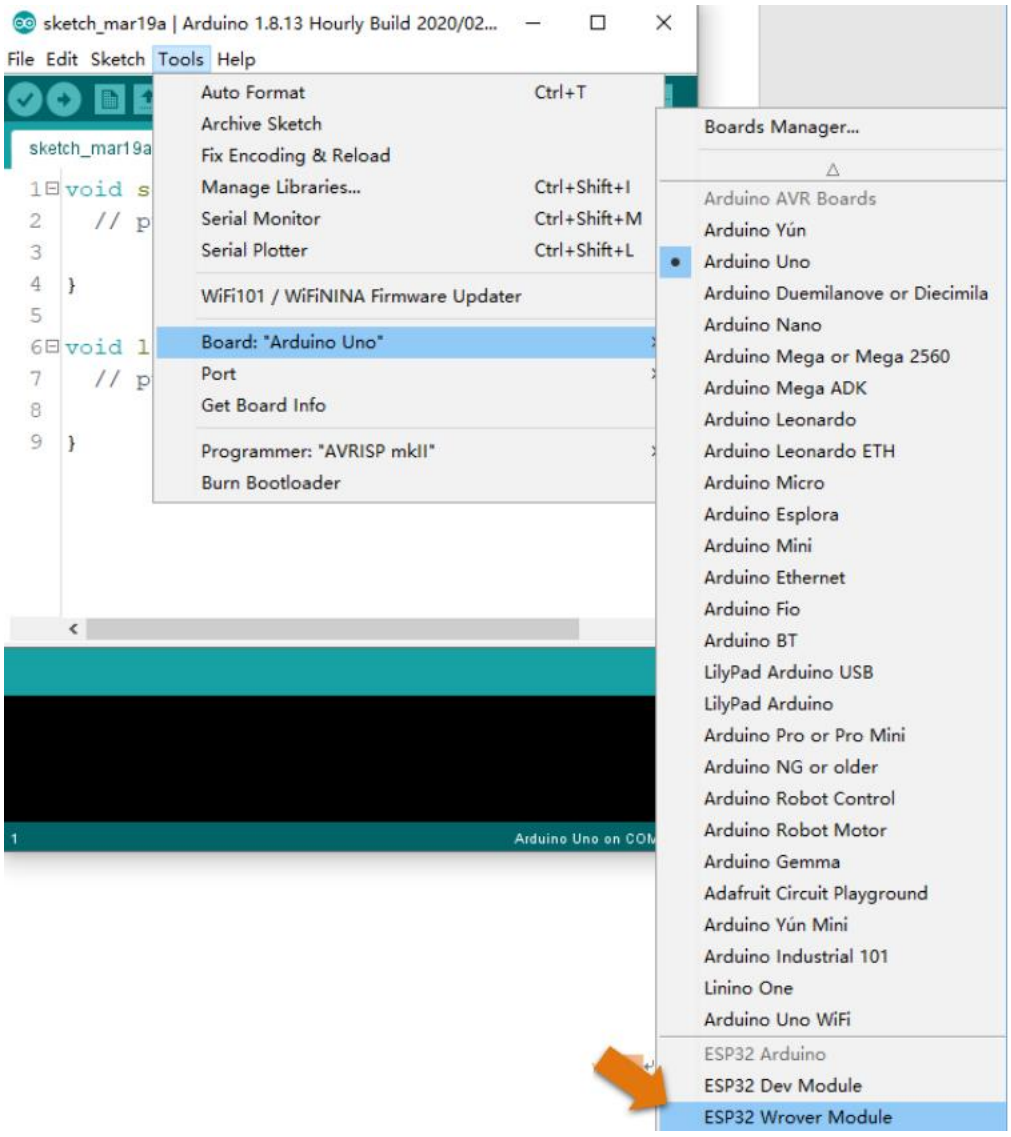

Figure 11: Verify EPS32 Wrover module is selected

3.1.4 Installing Python or Anaconda

For using the codes of Python, you need to install the Python or Anaconda which Python is included then, install dependency in Python in order to be able to run this code. To install any

library, you can first install pip then use pip install … to install any library you need. In our code, NumPy, Pandas, and Plotly will be used.

After installing Anaconda, you will see the image below.

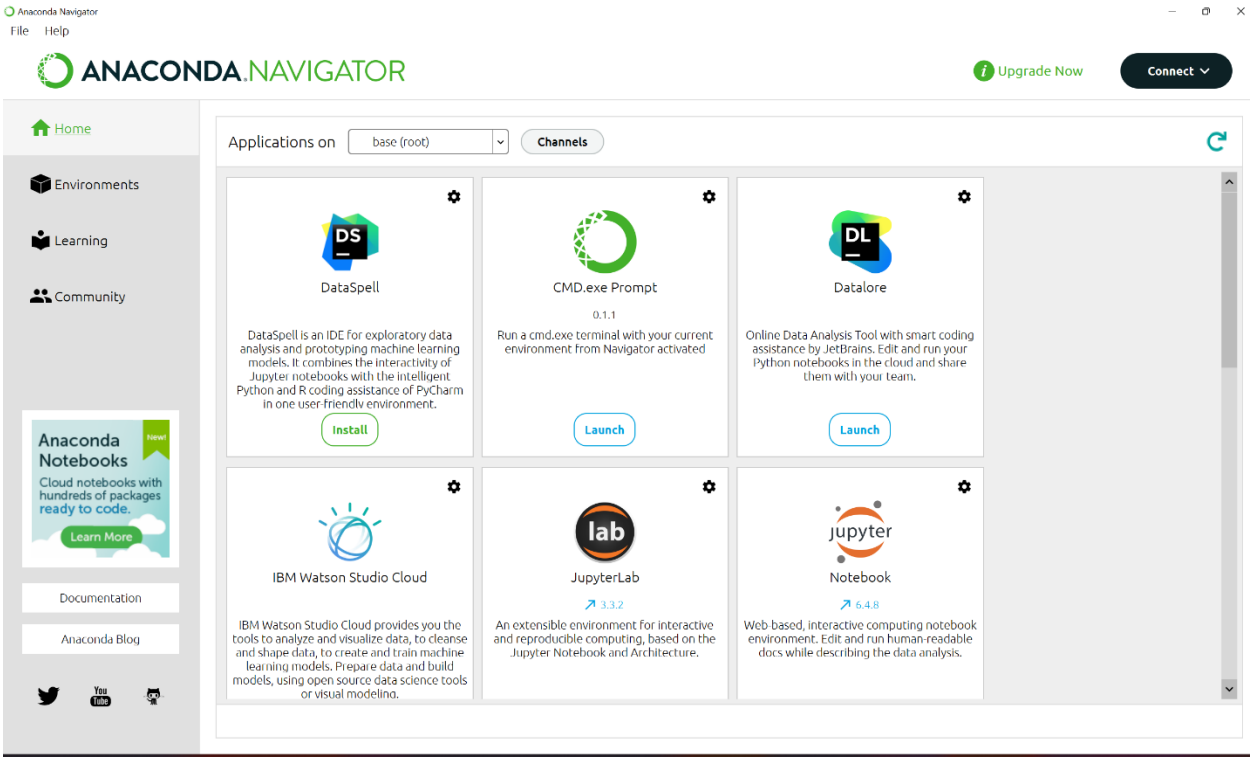

Figure 12: Anaconda Navigator

3.1.5 Installing Python Libraries

Install all necessary Python libraries which are NumPy, Pandas, and Pyplot. Later in the documentation, you will see how to use libraries and codes.

3.1.6 Installing PowerBI

To use Power BI, you also need to install Power BI on your computer or use your Microsoft account to have an online version. Once you open it you will see the image below.

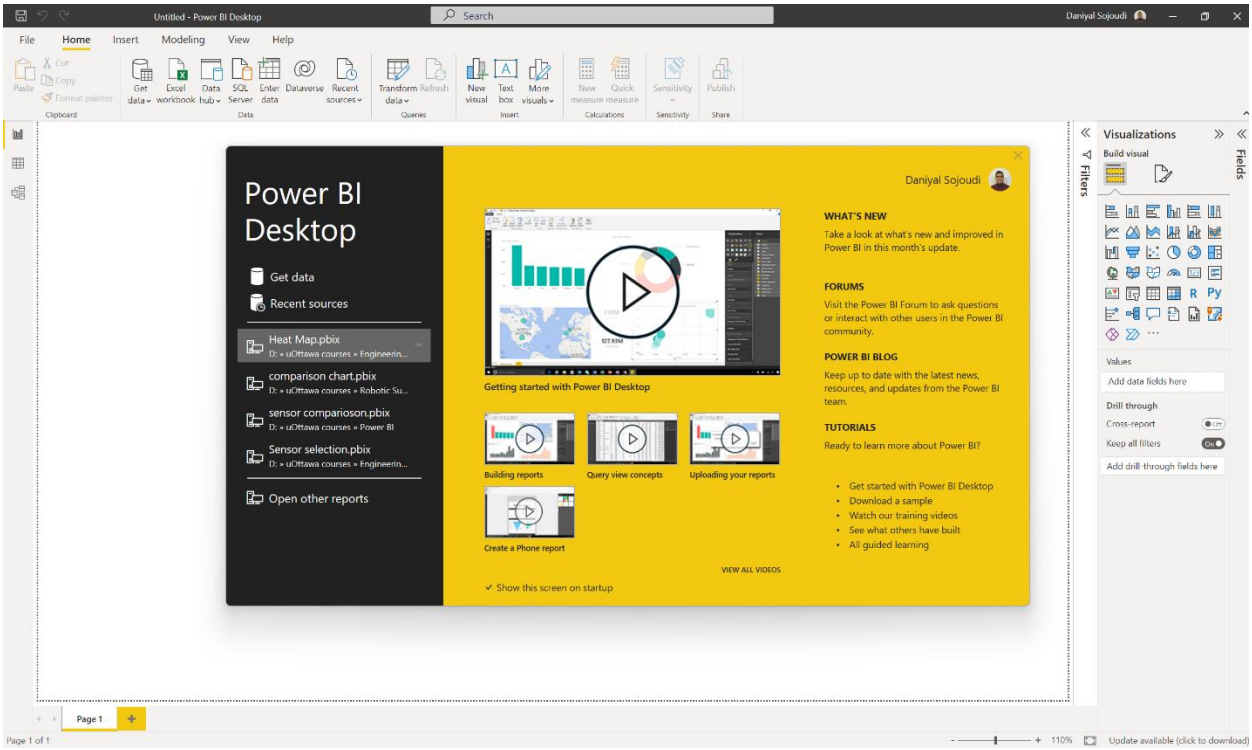

Figure 13: Installing Power BI

3.2 User Access Considerations

As of now, there are no restrictions for the users of our device. It is intended to be used by the public biking community who consent and are aware of the presence of a device that can track and log their location.

3.3 Accessing the System

3.3.1 Powering the Device

To power the device, simply connect the micro-USB port on the ESP32 board to a type A-USB port on your computer. This connection will power the entire system and allow for data transfer.

3.3.2 Installing Required Libraries

Several libraries are required to compile and upload our code or sketch. The libraries required are:

- Wi-Fi Library
- Hardware Serial Library
- TinyGPSPlus Library
- PMS Library
- MQ135 Library

To install these libraries, please follow these steps:

1) Click on the libraries' manager on the left side toolbar.

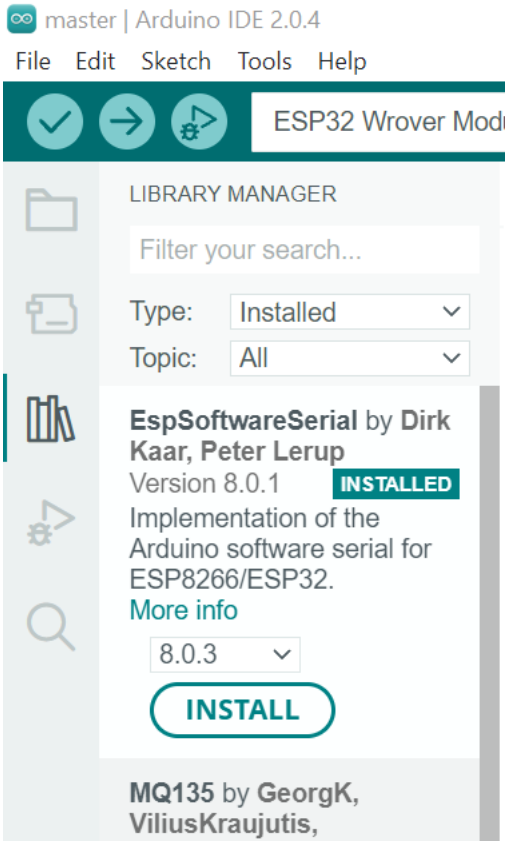

Figure 14: Library Manager

2) In the search bar, search for each one of the libraries listed above, and click on install one-by-one.

3.3.3 Uploading the Sketch

The code or sketch has already been programmed into the board. However, in case it gets reset, please follow the following procedures:

1) Ensure that the right communication port and board type are selected. Check the bottom right corner, you should be able to read “ESP32 Wrover Module on COMx”, x being the number of your port.

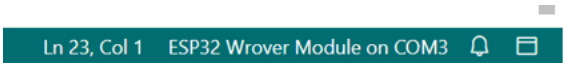

Figure 15: Ensure that the right communication port and board type are selected

2) Verify that the sketch is compiled without any errors by clicking on the “Verify” button.

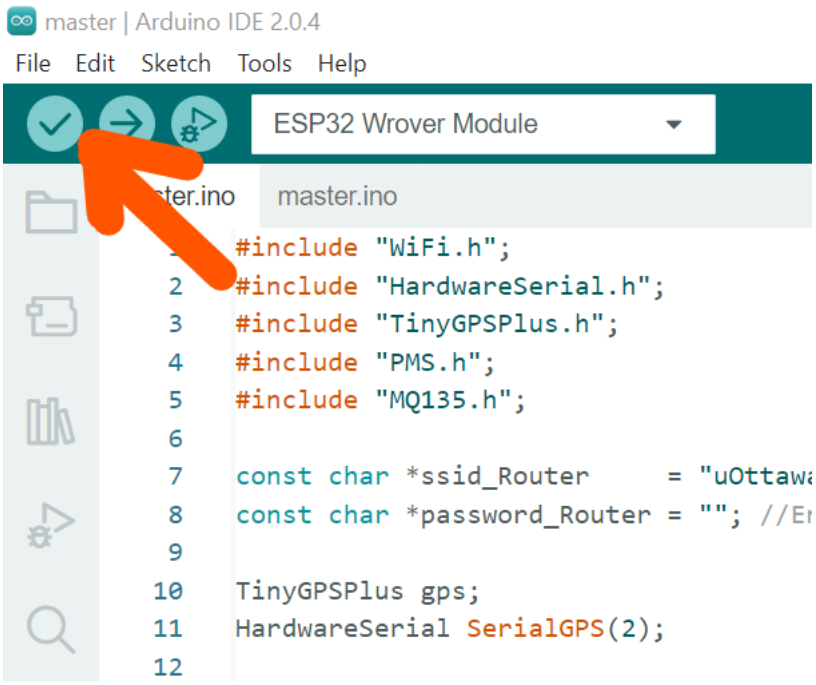

Figure 16: Compile sketch

3) After the sketch is compiled, upload it to the microcontroller by clicking on the "Upload" button.

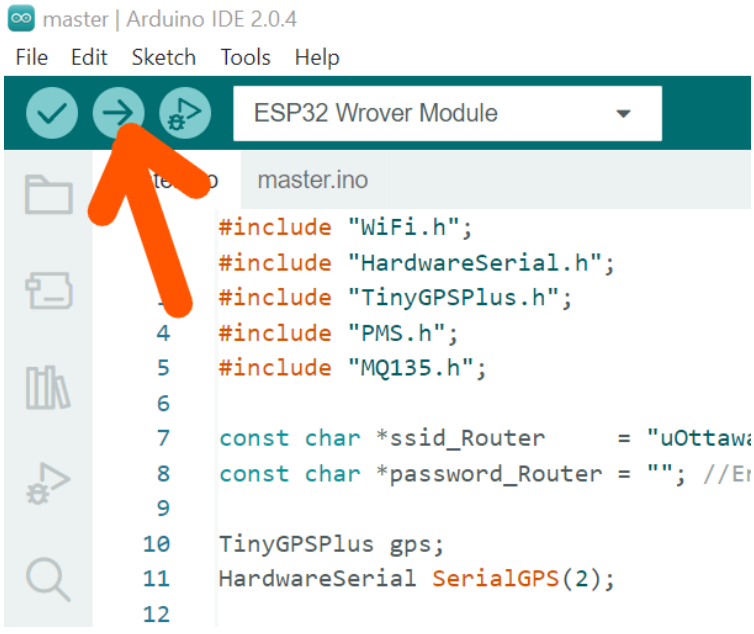

Figure 17: Upload the sketch to the microcontroller

4) Open the serial monitor by clicking on “Tools”, then “Serial Monitor”.

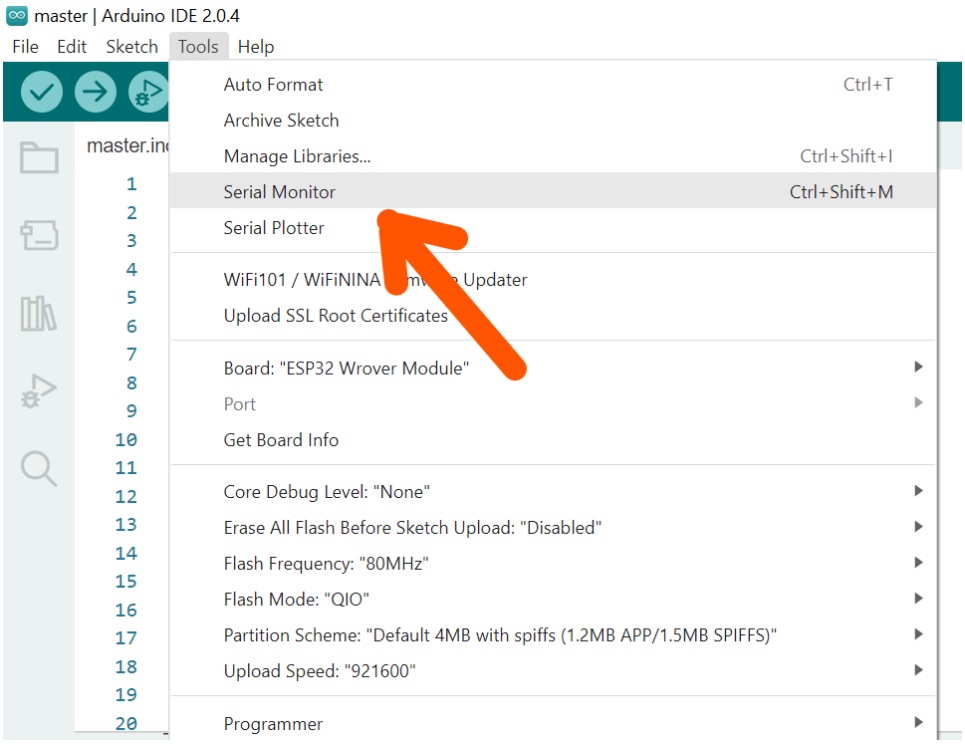

Figure 18: Open the serial monitor

5) In the serial monitor section, set the baud rate to “115200 baud”.

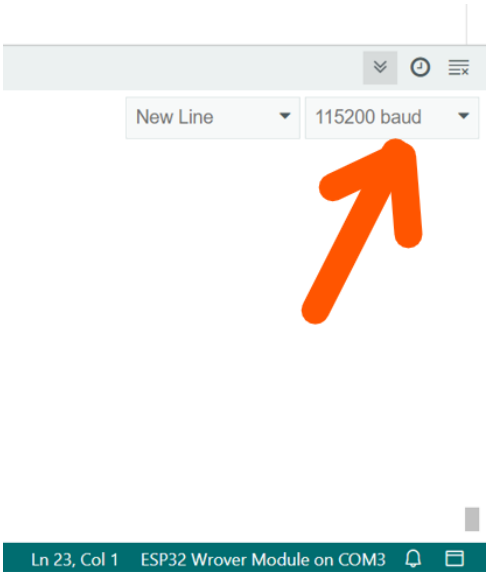

Figure 19: Setting the baud rate

6) Ensure that the values of the GPS coordinates, PMS and MQ135 sensors are displayed correctly in the serial monitor.

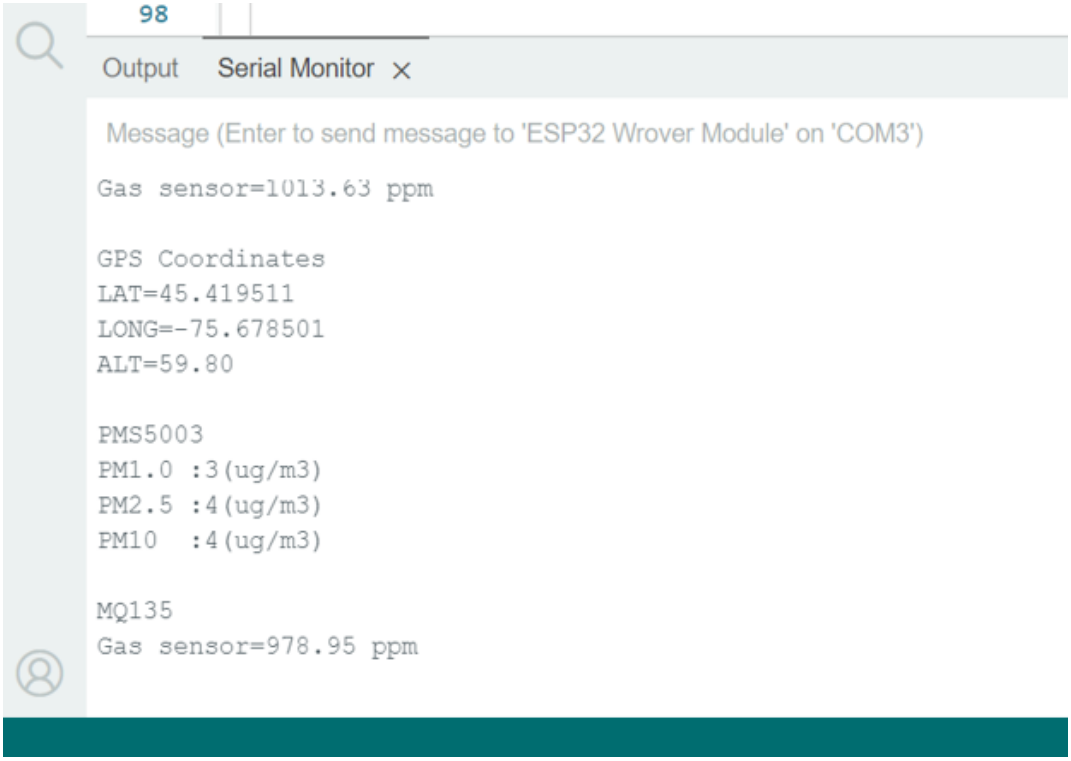

Figure 20: Serial monitor window

3.4 Exiting the System

Plugging off the device from the computer will make it stop immediately since it is the only power source. An external battery pack and storage module can also be integrated to keep the device running on the go.

If you are using Python for exiting, just save the file and it will be automatically saved in your Jupyter Notebook.

When you are using Power BI, you must follow the steps below to get output data.

1- Click on file on top left.

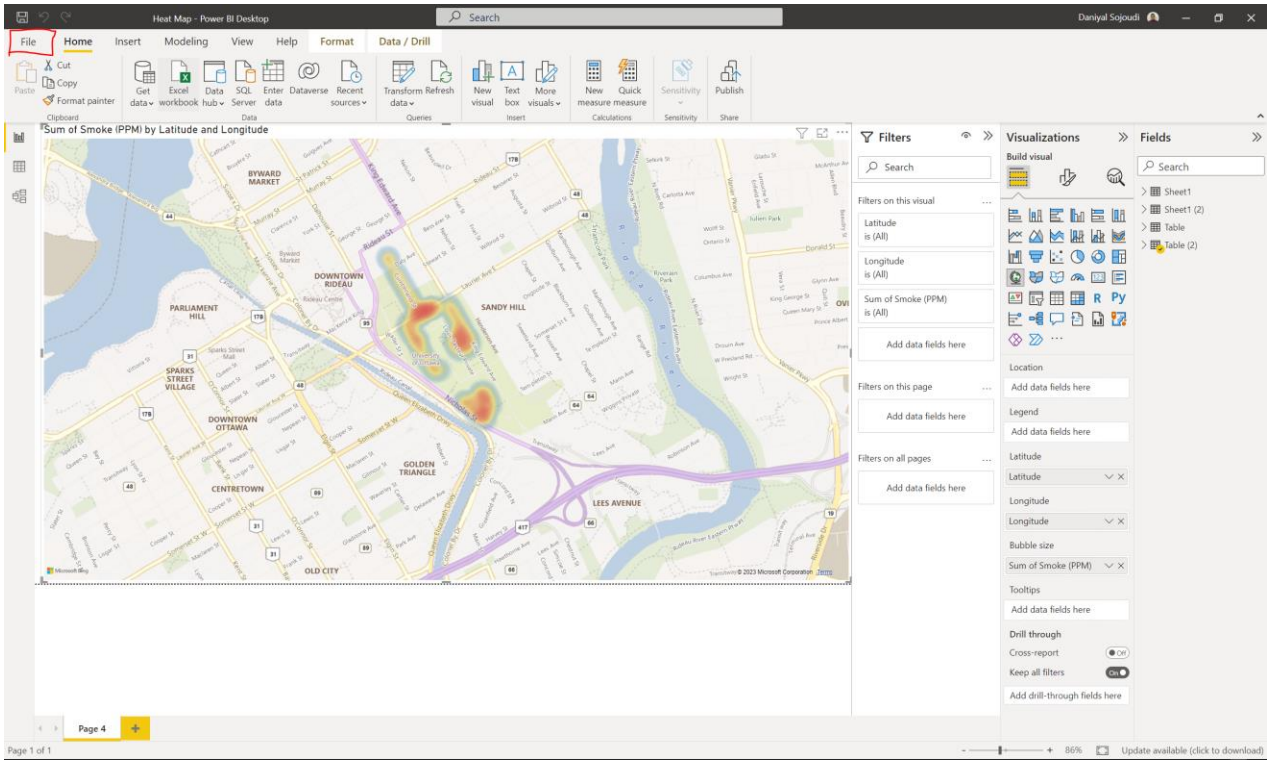

Figure 21: Get output data of Power BI (Step 1, Selecting File)

2- Click on export.

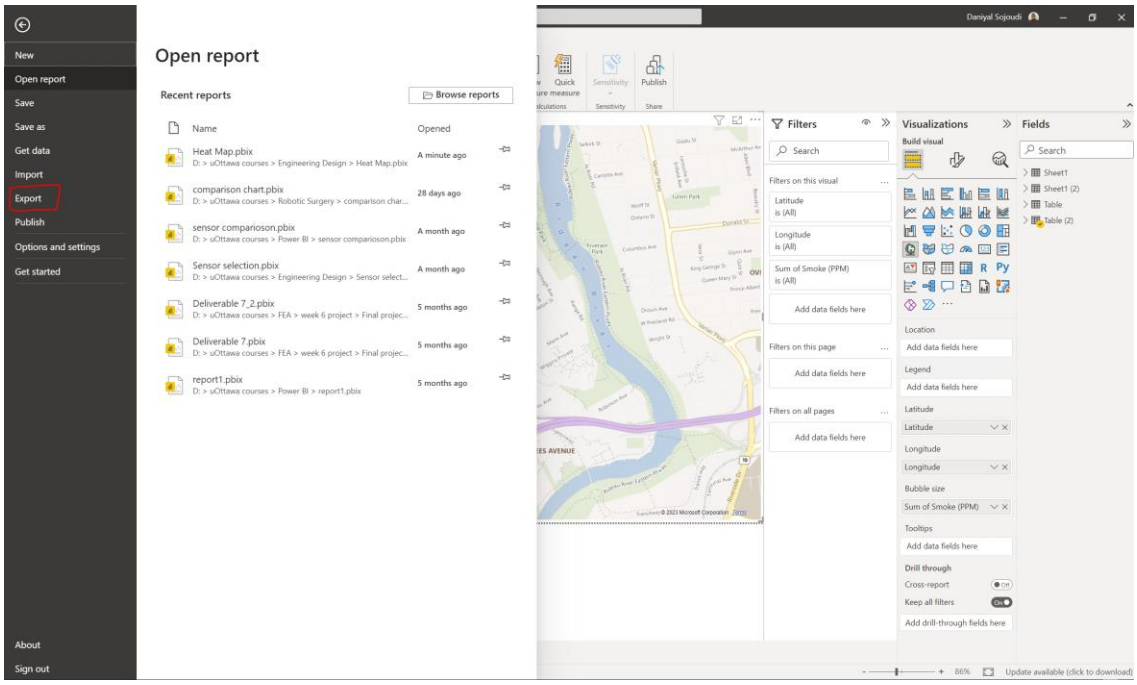

Figure 22: Get output data of Power BI (Step 2, Selecting Export)

3- Export as a pdf. Then save the file itself and close the software.

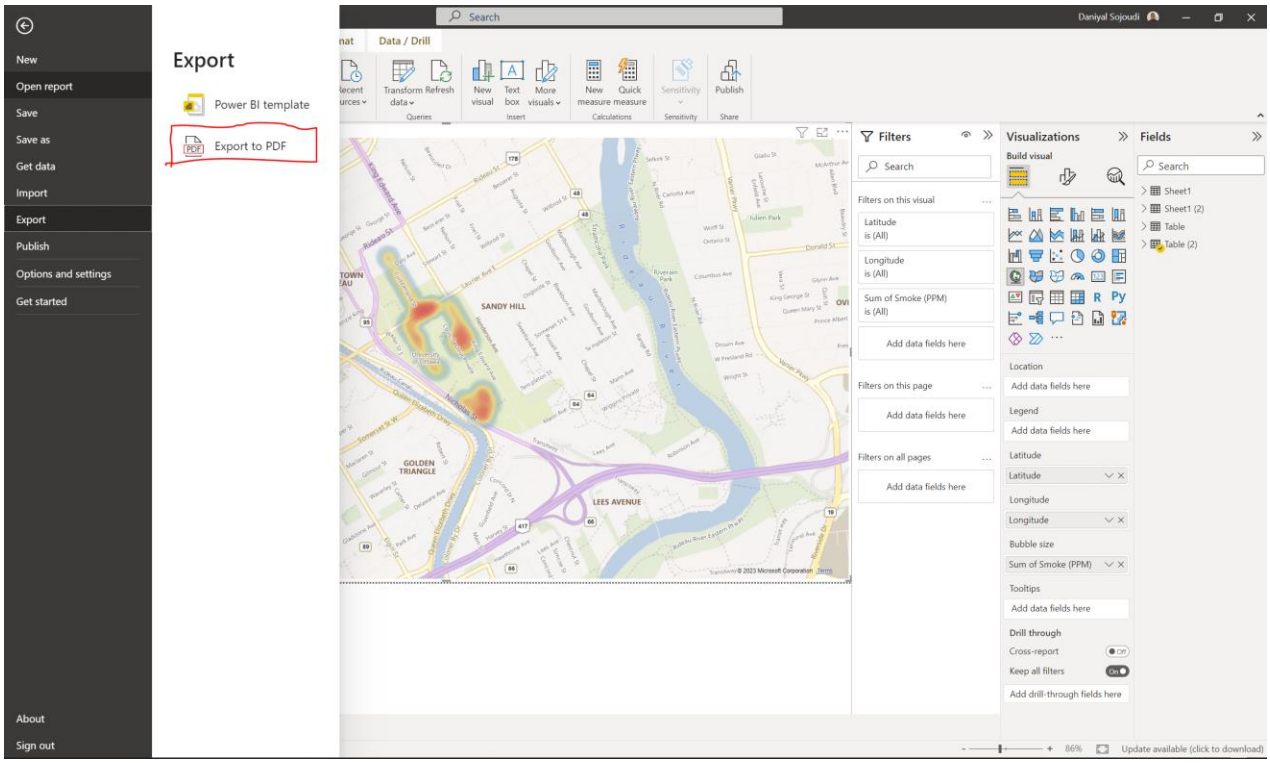

Figure 23: Export visualization as a pdf

4 Using the System

4.1 Exporting the Data

To export the data from Arduino IDE software to an external datasheet, such as an Excel workbook, please follow the steps below:

1) Condition the data by removing any strings and characters associated with the serial output with a new line at the end of each reading. Follow the format similarly to the picture below:

```
Formatting output for Excel DataStream
Serial.print(gps.location.lat(), 6);  //lat
Serial.print(",");
Serial.print(gps.location.lng(), 6);  //long
Serial.print(",");
Serial.print(val1); //pm1.0
Serial.print(",");
Serial.print(val2); //pm2.5
Serial.print(",");
Serial.print(val3); //pm10
Serial.print(",");
Serial.print(correctedPPM); //mq135
Serial.print(",");
Serial.print(sensor_Aout); //mq135
Serial.print(",");
Serial.println();
```

Figure 24: Formatting data for Data streaming

2) Open Excel and enable the Data Streamer add-in by going to “Files”, then “Option” then “Add-Ins”. Select “COM Add-Ins” in the “Manage” box then click “Go”. A new dialog box appears, in which you have to search for “Microsoft Data Streamer for Excel” and make sure that the box is ticked. Finally, click “Ok” and close all the additional windows.
3) Make sure to close Arduino IDE before proceeding to the next step.
4) Head to the new “Data Streamer” tab that appears in Excel. Click on “Connect a Device” and choose the corresponding COMx port that was chosen previously in Arduino IDE.

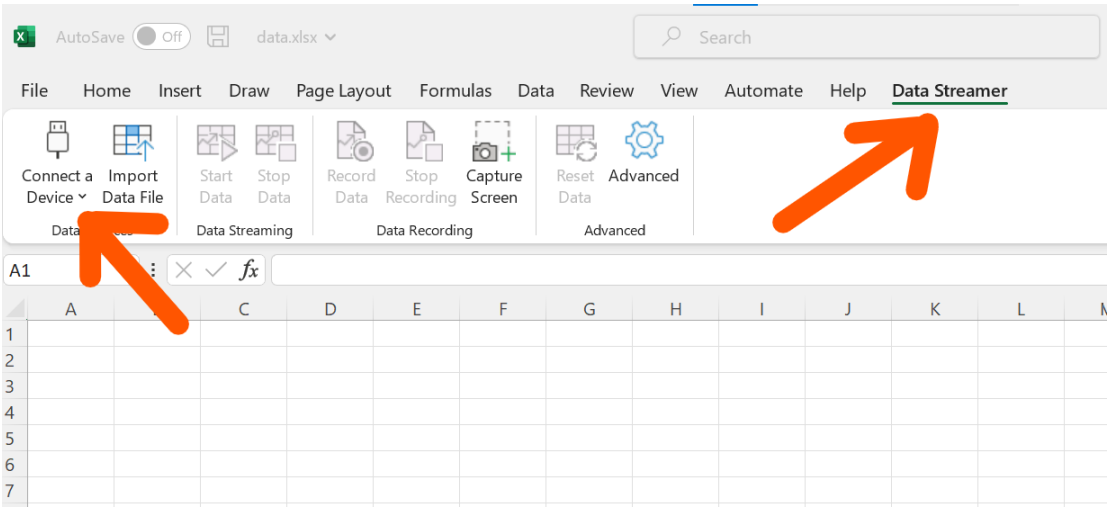

Figure 25: Connecting the device to Data Streamer

5) Make sure the correct baud rate is selected by going to “Advanced”, “Settings” then “Baud Rate”.
6) Click on “Start Data” to start recording the data in real-time to the excel worksheet.
7) After collecting sufficient data, click on “Stop” and make sure to save the file.

4.2 Visualization using Python

1) Open the anaconda or Python IDE.

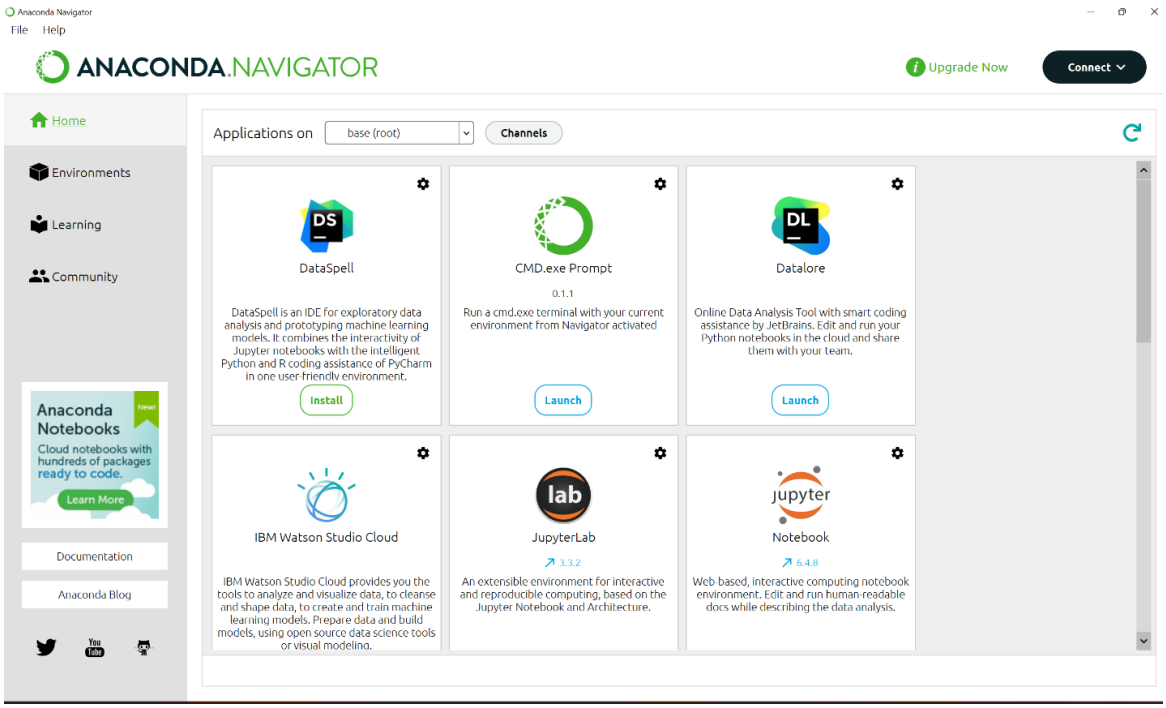

Figure 26: Interface of Anaconda

2) Click on Jupyter Notebook: you will see the page below then you can click on new to create a new notebook to write in it.

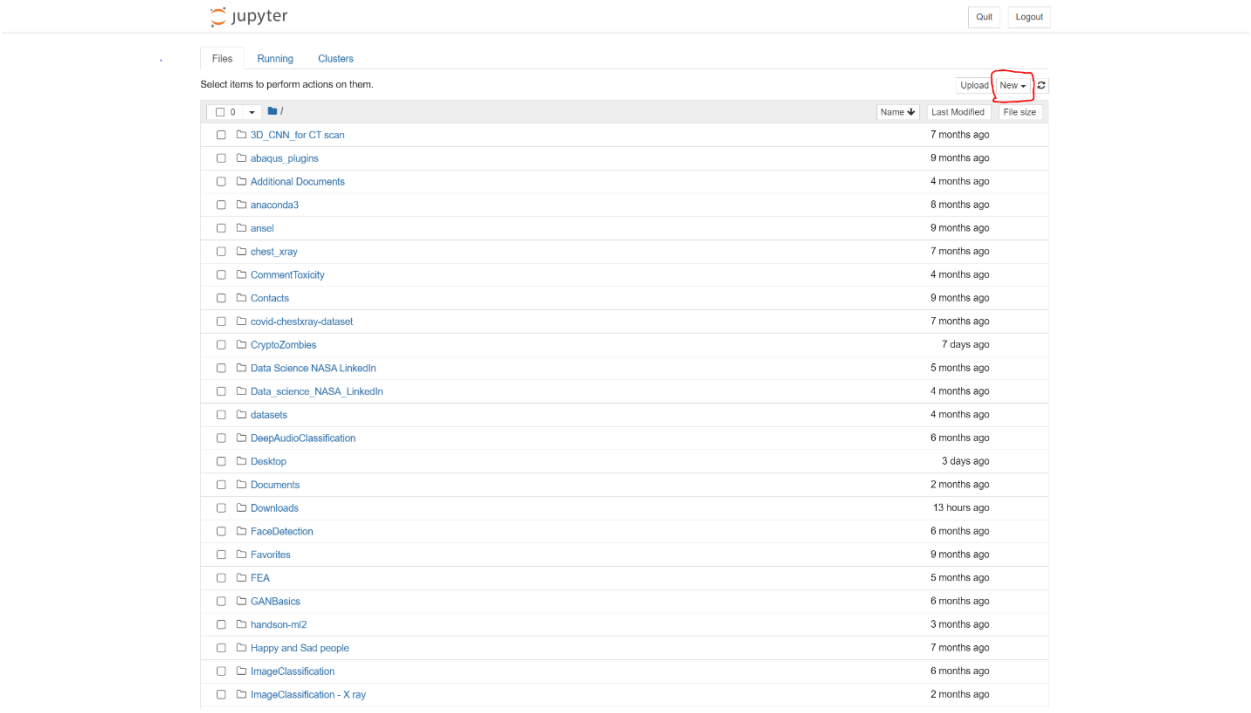

Figure 27: Create a new file (Step 1)

3) Click on Python 3 (ipykernel) to create a new kernel.

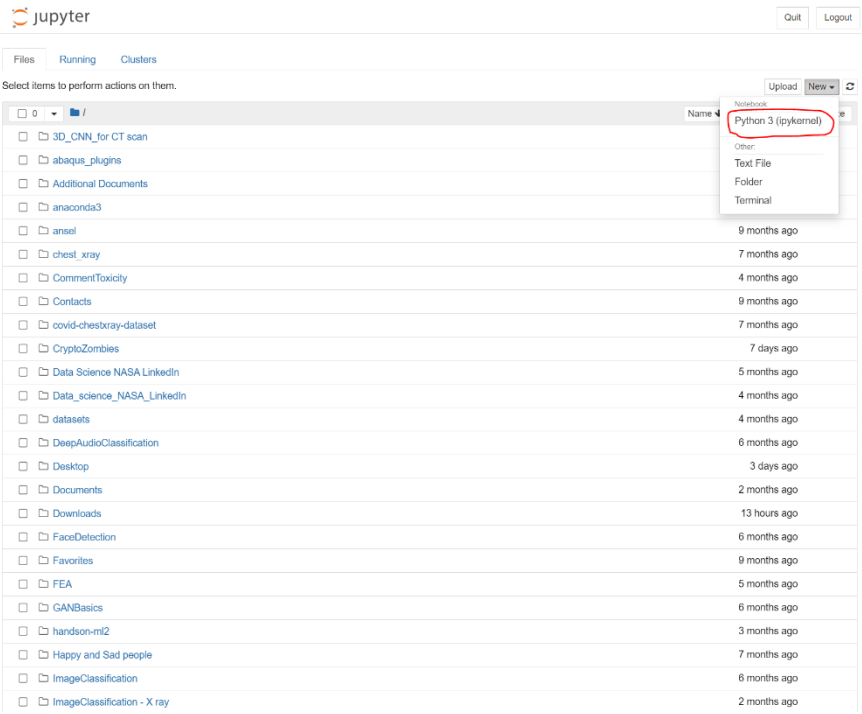

Figure 28: Create a new file (Step 2)

4) In the next step you will see a page that you can start coding and if you click on the red box, you will be able to change the name of the kernel.

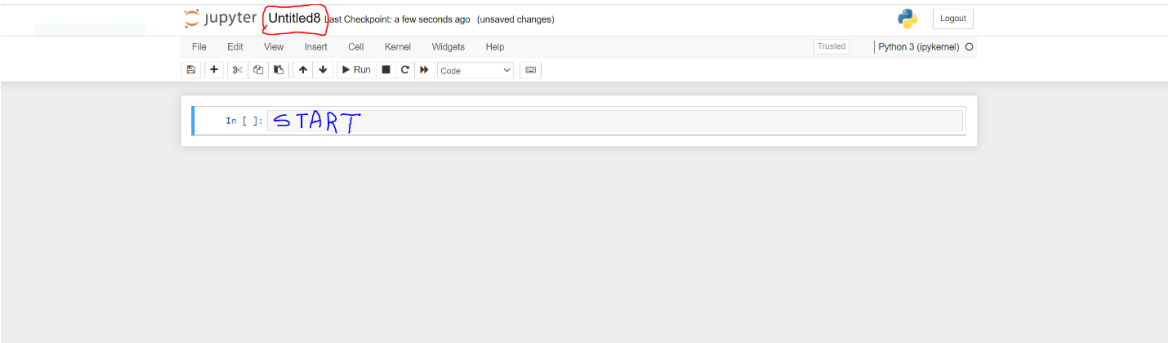

Figure 29: Kernel interface to start coding

5) Enter the code as you can see in this picture then you will be able to load pollution data in the live map. [1] import Pandas after installing then read the Excel file from the experiment.[2] make the table visible.[3] install Pyplot.[4] use Pyplot and other libraries to plot data.

- Note 1: be careful about the location of the Excel output in your computer and you must give the local address of the file in your computer.
- Note 2: you need to install dependency in Python in order to be able to run this code to install any library you can first install pip then use pip install … to install any library you need. In our code, NumPy, Pandas, and Plotly will be used.
- Note 3: you may need to restart the kernel to make the code work.

In [1]:
```
import pandas as pd

df = pd.read_excel(r'D:\uOttawa courses\Engineering Design\Data collection_new.xlsx')
```

In [2]:
```
df.head()
```

Out[2]:

	Latitude	Longitude	Smoke (PPM)
0	45.427202	-75.686802	4990.17
1	45.427202	-75.686802	7831.08
2	45.427202	-75.686802	5528.97
3	45.427202	-75.686802	9492.87
4	45.427202	-75.686802	7549.88

In [3]:
```
pip install plotly==5.8.0
```

```
Requirement already satisfied: plotly==5.8.0 in c:\users\daniyel\anaconda3\lib\site-packages (5.8.0)
Requirement already satisfied: tenacity>=6.2.0 in c:\users\daniyel\anaconda3\lib\site-packages (from plotly==5.8.0) (8.0.1)
Note: you may need to restart the kernel to use updated packages.
```

```
WARNING: Ignoring invalid distribution -pencv-python-headless (c:\users\daniyel\appdata\roaming\python\python39\site-packages)
WARNING: Ignoring invalid distribution -pencv-python-headless (c:\users\daniyel\appdata\roaming\python\python39\site-packages)
WARNING: Ignoring invalid distribution -pencv-python-headless (c:\users\daniyel\appdata\roaming\python\python39\site-packages)
WARNING: Ignoring invalid distribution -pencv-python-headless (c:\users\daniyel\appdata\roaming\python\python39\site-packages)
WARNING: Ignoring invalid distribution -pencv-python-headless (c:\users\daniyel\appdata\roaming\python\python39\site-packages)
WARNING: Ignoring invalid distribution -pencv-python-headless (c:\users\daniyel\appdata\roaming\python\python39\site-packages)
```

In [4]:
```
import pandas as pd
import numpy as np
import plotly.express as px
fig = px.density_mapbox(df, lon='Latitude', lat='Longitude', z='Smoke (PPM)',
                        mapbox_style="stamen-terrain")

fig
```

Figure 30: Python code for visualization

Figure 31: Visualization using Python

4.3 Visualization using Power BI

1) Install and open Power BI.

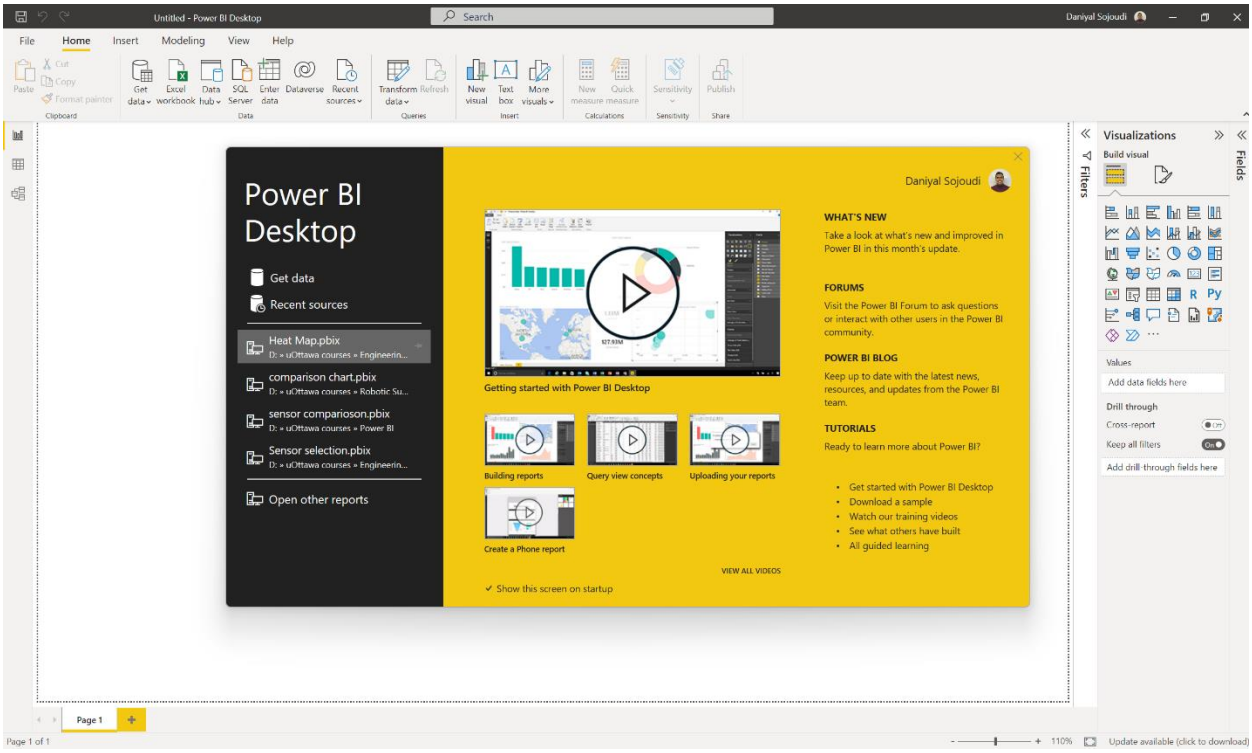

Figure 32: Power BI interface

2) Since we have Excel file to import click on the Excel as shown in the box.

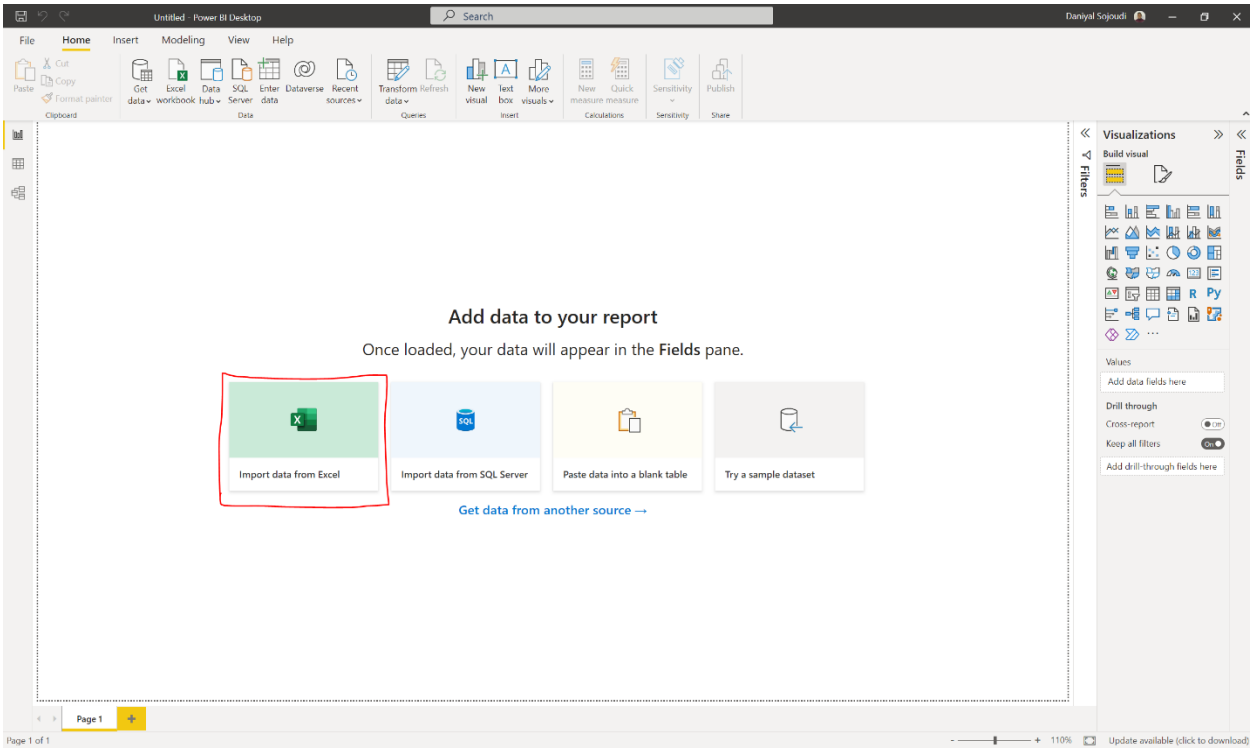

Figure 33: Choose Excel as an input

3) Select the data collected excel file in your computer.

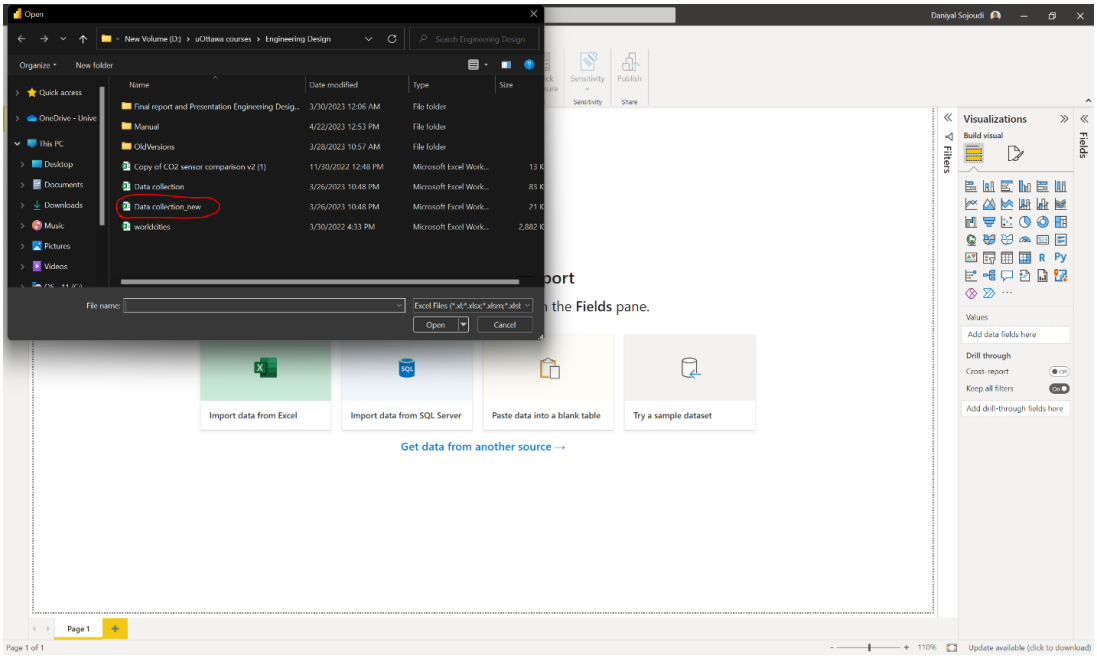

Figure 34: Choosing the Excel file

4) Check the sheet 1.

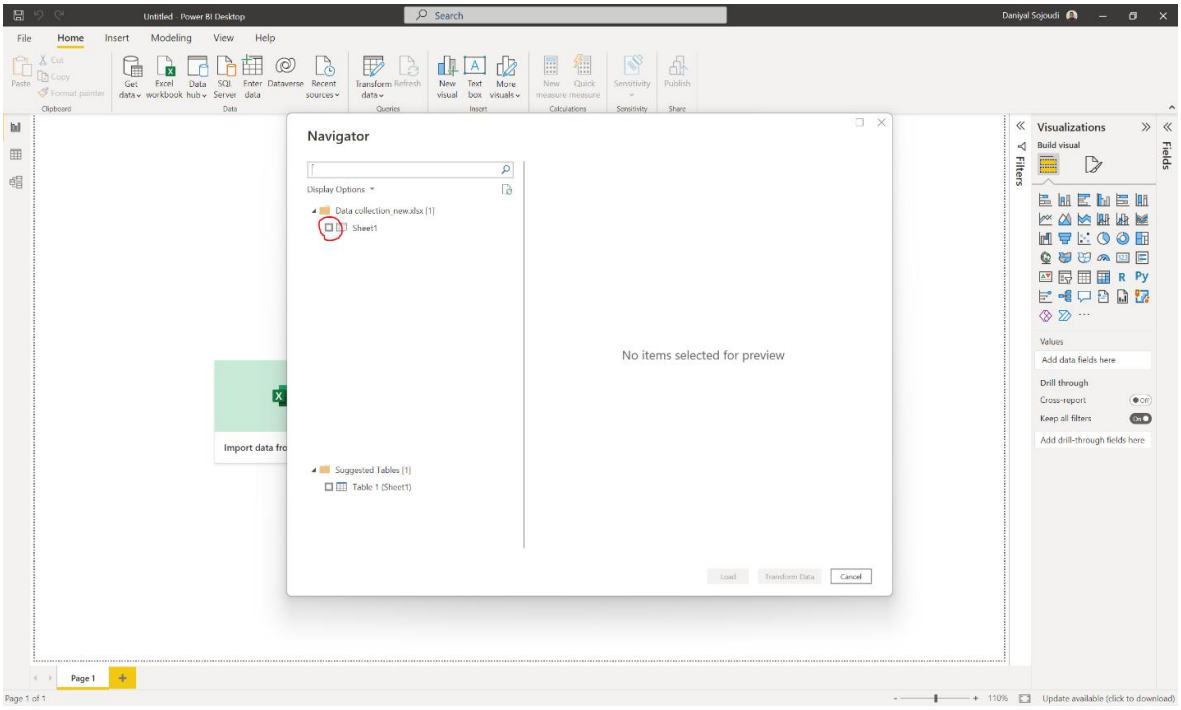

Figure 35: Choosing the file that we have

5) Then you will see the excel file and you can load it in the data base.

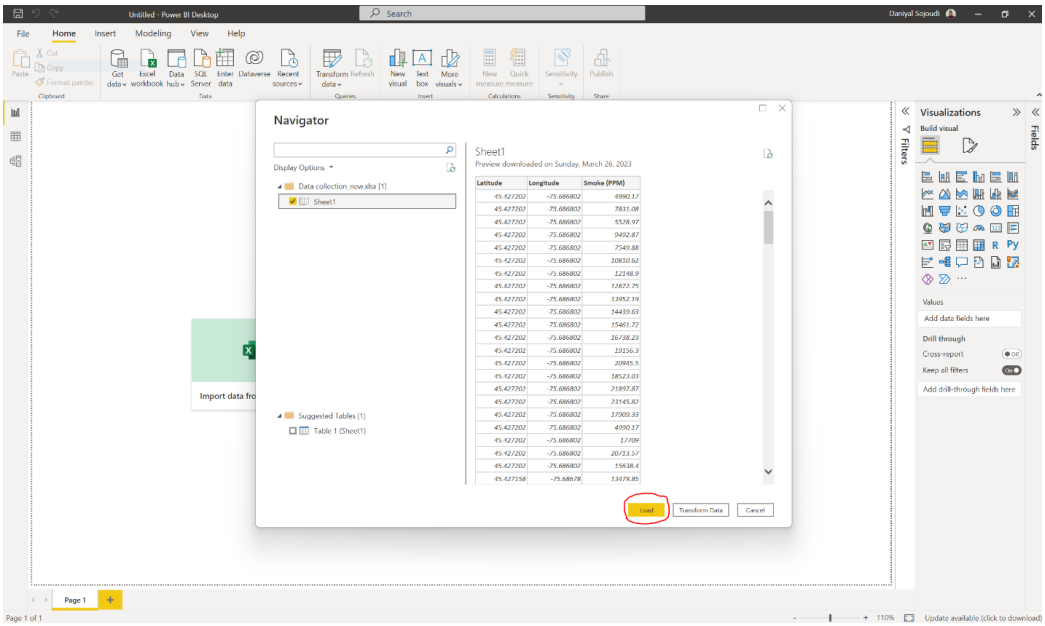

Figure 36: Load pollution data

6) Now you can see your datasheet on the right and different visualization options one of which is map as shown.

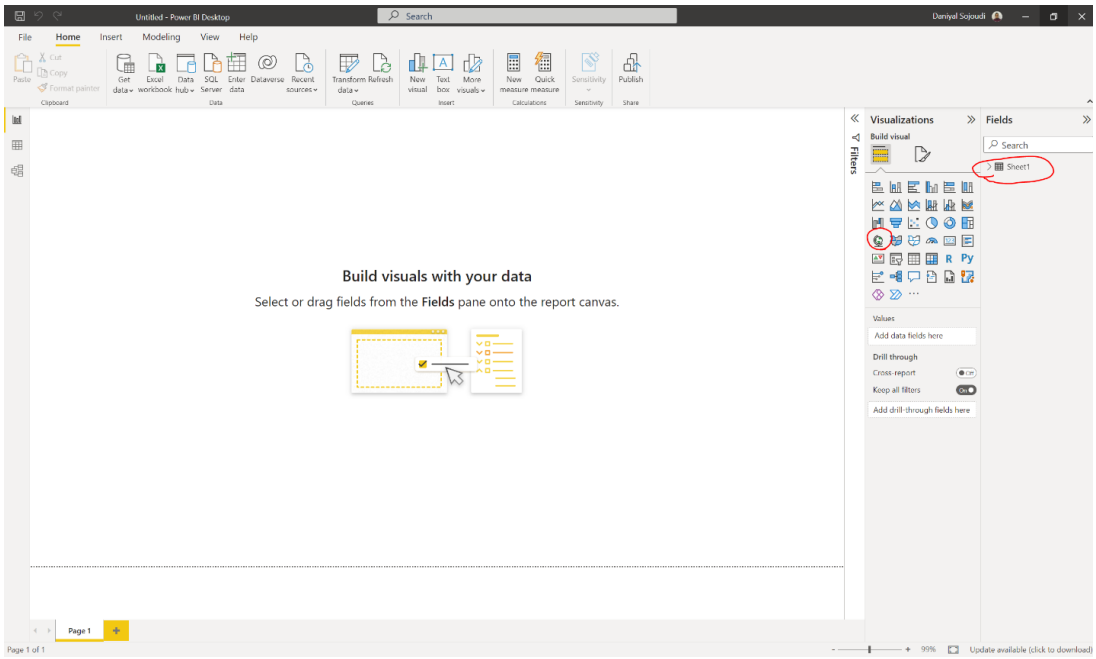

Figure 37: Choosing the file and Map option in the power BI for live Map

7) In your sheet, you select Latitude and Longitude and import it in the map requirements then you will see a live map with the data points.

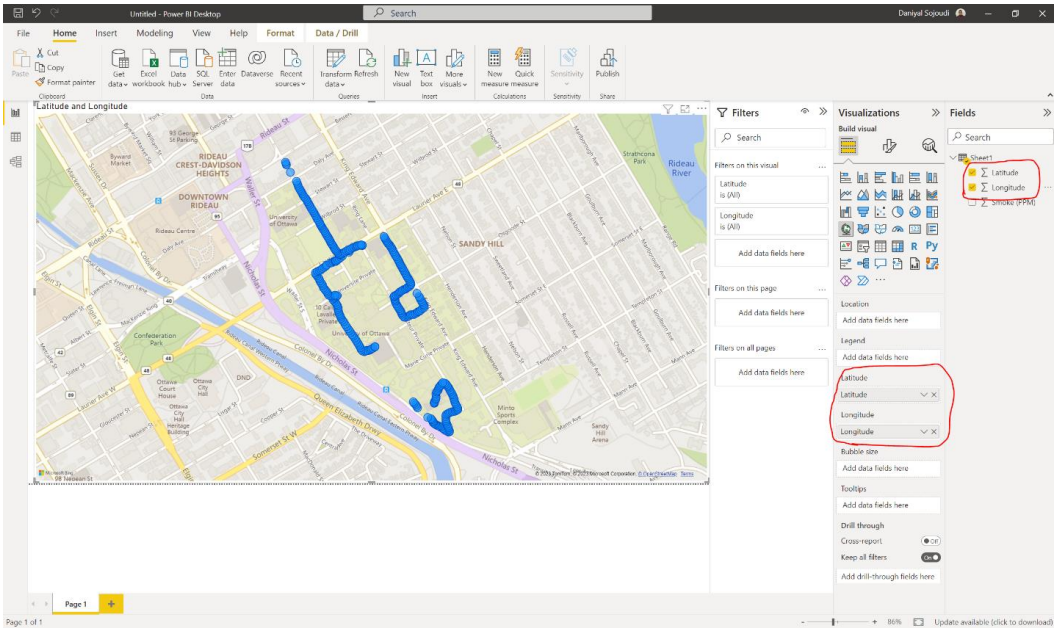

Figure 38: Choosing Latitude and Longitude to have a live map

8) In the format visualization you can customize and create a heatmap.

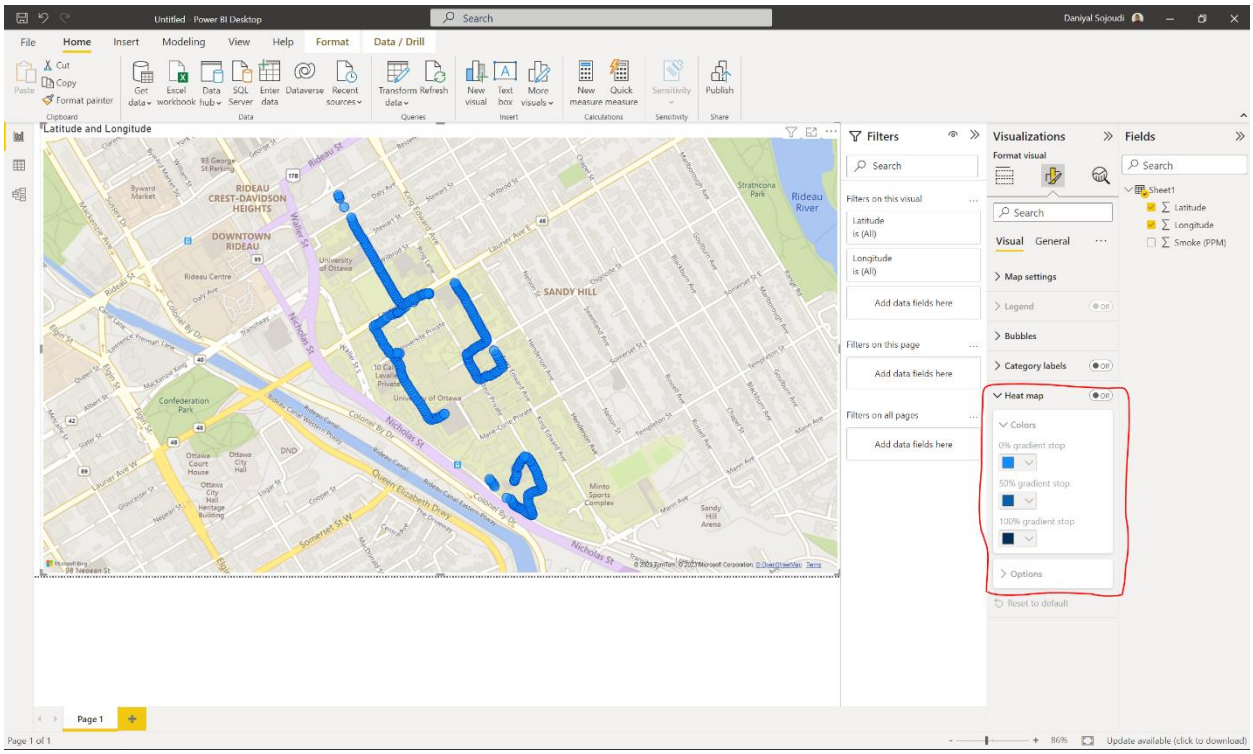

Figure 39: Customized a heat map

9) You can build a customized heat map like below.

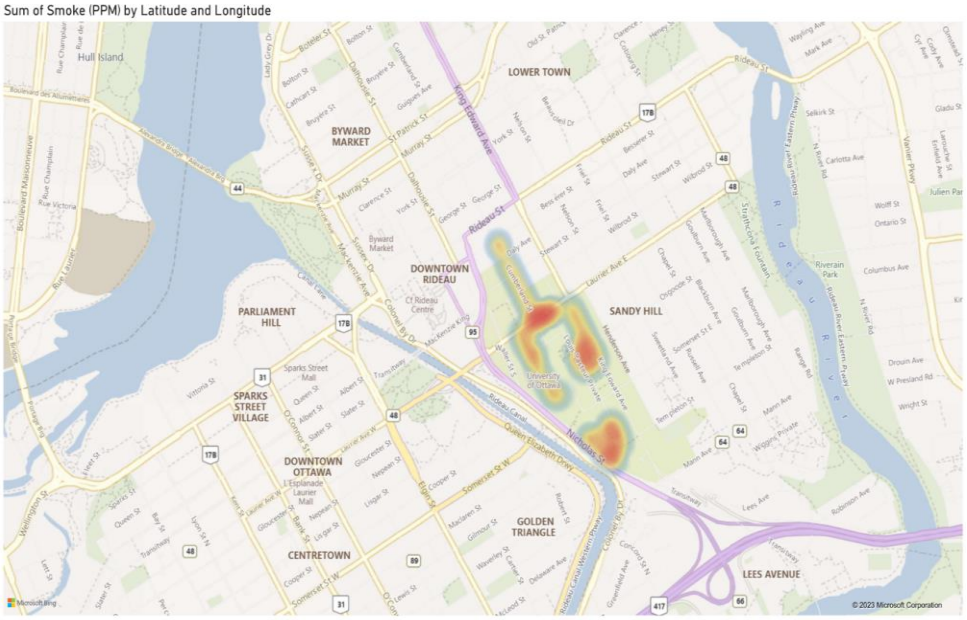

Figure 40: Final heat map

5 Troubleshooting & Support

5.1 Error Messages or Behaviors

Failure to Connect to GPS:

The device may fail to connect to GPS or experience weak signal strength. This may occur if the GPS antenna is obstructed, if there is a problem with the GPS receiver, or if the device is in an area with poor GPS coverage. The corrective action may involve repositioning the device, testing the GPS receiver, or relocating to a different area with better GPS coverage.

Failure to Upload Data:

The device may fail to upload pollution and GPS data to the cloud. This may occur due to a network connectivity issue or if there is a problem with the cloud server. The corrective action may involve checking network connectivity, resetting the device, or contacting the cloud service provider for support.

Data Corrupted During Upload:

Data corruption can occur if there is a disruption in network connectivity during the upload process or if there is a problem with the device's memory. The corrective action may involve re-uploading the data or replacing the device's memory.

Physical Damage:

The device may be prone to physical damage due to its exposed design. The device may become damaged due to exposure to the elements or accidental drops. The corrective action may

involve protecting the device with a case or covering or repairing the device if it becomes damaged.

Software Glitches:

The device may experience software glitches, which can cause errors or unexpected behavior. This may occur due to bugs in the device's software or if the software is out of date. The corrective action may involve updating the software or contacting the device manufacturer for support.

Insufficient Memory:

The device may run out of memory if it collects a large amount of data over time. This can cause the device to slow down or crash. The corrective action may involve increasing the device's memory capacity or deleting old data to free up space.

Malfunctioning Sensors:

The device's pollution sensors may malfunction, leading to inaccurate data collection. This can occur due to sensor damage or wear and tear. The corrective action may involve replacing the sensors or calibrating them to ensure accurate data collection.

Python:

As a programming language has its own challenges but if you follow the instructions provided in the manual including installing library, and Python itself properly it will be fine. Regarding the Power BI there is no challenge because the software is very user friendly.

5.2 Special Considerations

- Make sure all cables and connections are properly seated and secured if the AQMS is not collecting data.
- If the AQMS is not communicating with a laptop or other data storage device, check the Wi-Fi connection and ensure that the software is up-to-date and properly configured.
- Check the sensors to ensure that they are calibrated properly and free of debris or other obstructions if the AQMS is producing inaccurate readings or data that is out of range. Check the air flow to ensure that the device is not being exposed to excessive dust or other airborne particles if hydrocarbon data is out of range.
- Check the GPS connection and ensure that the device is in an area without tall buildings or other obstructions that could interfere with the signal if the GPS isn't working.
- Check the software and firmware versions for updates or known issues if the AQMS does not respond to commands or seems to be malfunctioning.
- Consider whether weather conditions, proximity to other sources of pollution, or interference from nearby electronics could be affecting the AQMS if it generates unusual or unexpected results.

5.3 Maintenance

To ensure the AQMS prototype works properly and is accurate, it must be maintained regularly. The following maintenance tasks should be performed on a regular basis:

1. AQMS sensors should be calibrated periodically to ensure accurate readings. The calibration frequency will depend on the specific sensors and the prototype's operating conditions.
2. Inspection of hardware components: The AQMS should be inspected regularly for signs of wear or damage. This includes checking cables and connections for fraying or other damage, as well as inspecting the housing for cracks or other signs of damage.

3. Cleaning of sensors and housing: The AQMS should be cleaned regularly to prevent dust buildup that could interfere with the sensors' accuracy. The sensors and housing can be cleaned with a soft cloth or brush, and in some cases, compressed air can be used to remove stubborn debris.
4. Data backup: Regular backups of collected data should be performed to prevent the loss of important information in the event of a failure or other issue.

5.4 Support

- Waleed Qanbar (mw.qanbar@uottawa.ca) is in charge of integrating the microcontroller system and collecting the data.
- Daniyal Sojoudi (daniyal.sojoudi@uottawa.ca) specialized in data analysis and visualization.

6 Product Documentation

6.1 BOM (Bill of Materials)

Table 2: Bill of Materials

Item name	Description	Units of measure	Quantity	Unit cost	Extended cost	Link
Microcontroller Board & Starter Kit	ESP32 Wrover with built in WiFi Module. Includes wires and breadboard.	Unit	1	$30.95	$30.95	https://www.amazon.ca/dp/B09BC1N9LL?psc=1&ref=ppx_yo2ov_dt_b_product_details
GPS Module	Beitian BN-180	Unit	1	$25.96	$25.96	https://www.amazon.ca/dp/B086ZKWYP5?psc=1&ref=ppx_yo2ov_dt_b_product_details
PM2.5 Sensor	PMS5003	Unit	1	$47.45	$47.45	https://www.amazon.ca/dp/B09BQZP2CY?psc=1&ref=ppx_yo2ov_dt_b_product_details
Hydrocarbons & Smoke Sensor	MQ-135	Unit	1	$9.70	$9.70	https://www.amazon.ca/dp/B09K68TZTG?psc=1&ref=ppx_yo2ov_dt_b_product_details
Total product cost (without taxes or shipping)						$114.06
Total product cost (including taxes and shipping)						$127.62 + $0 shipping

6.2 Equipment list

1- Microcontroller Board & Starter Kit → ESP32 Wrover with built in WiFi Module. Includes wires and breadboard

2- GPS Module → Beitian BN-180

3- PM2.5 Sensor → PMS5003

4- Hydrocarbons & Smoke Sensor → MQ-135

6.3 Testing & Validation

While collecting data, various factors were noticed that might have impacted the data quality. Firstly, it was observed that the GPS device utilized in the project was inexpensive and experienced malfunctions close to tall buildings, which were abundant in the area. This could have affected the accuracy of the data obtained and ought to be taken into account for upcoming projects. Secondly, it was noted that there was less smoke around the University of Ottawa due to the low number of vehicles in the region. However, whenever cars went by, the smoke level increased, and when buses were present, the smoke level was found to increase even more, indicating that vehicular traffic significantly contributes to the pollution levels in the area. Thirdly, during the data collection period, it was windy, which might have affected the data collected since wind can disperse pollutants, causing it challenging to obtain precise readings. Therefore, when evaluating the outcomes, the impact of the wind on the data should be considered.

7 Conclusions and Recommendations for Future Work

The project involved constructing a portable air pollution measuring system that successfully utilized MQ135 and PMS 5003 sensors in conjunction with Wi-Fi and GPS modules to communicate with a microcontroller board. All components operated efficiently, and data was gathered from the University of Ottawa and the Rideau Street neighborhood. The collected data was visualized it using Python and Power BI, displaying a Heat Map of live pollution levels and a filter for selecting the least polluted route. The project's scalability, functionality, usability, and quality were all examined based on the process and outcomes. In the future, additional pollution sensors can be added to the system, and the map can be included in an application that suggests the least polluted route. Policymakers may utilize the data to comprehend how different populations are exposed to pollutants.

Lessons Learned:

1- Prototyping: Obtained experience in developing the device's hardware and software prototypes.

2- Sensor Selection: learned about several sensor kinds and how they may be used to measure air pollution.

3- Data analysis: acquired expertise in data collection and analysis to gain knowledge about patterns and trends in air pollution.

4- Integration: learned how to integrate various hardware parts to function as a system as a whole.

5- Project Management: gained expertise in project management, including formulating objectives, creating a schedule, and organizing team members.

Recommendations for future work:

1- Refine the sensor calibration: For accurate air pollution measurements, the sensors' calibration

is essential. In order to reduce measurement errors, future work should concentrate on optimizing the calibration procedure.

2- Expand the data collection: More data will be gathered, which will boost the analysis's and visualization's accuracy. The network of devices may need to be expanded in future work in order to collect data from other areas.

3- Improve data analysis techniques: Complex patterns in data on air pollution can be found using sophisticated data analysis techniques like machine learning. To learn more about the patterns and trends of air pollution, future research could concentrate on investigating these strategies.

4- Enhance the user interface: The device's user interface might be made more intuitive and user-friendly. The development of the device's interface could be the focus of future studies.

5- Integrating additional sensors: Additional sensors might be added to measure other air contaminants such as carbon monoxide, ozone, and nitrogen dioxide.

6- Developing a mobile app: may create an application for mobile devices that would enable people to monitor real-time air pollution data and get notifications when pollution levels are high.

7- Conducting field tests: might carry out testing in the field to confirm the precision of the instrument's measurements and find any potential problems.

8- Conducting outreach: might carry out outreach to neighbourhood community organizations and policymakers to encourage the usage of the device and its potential.

Overall, by providing precise and easily accessible data on air pollution, the project has the potential to help create a more sustainable future. By improving the device's design and enhancing its features, one can build on previous progress and keep having a positive influence on the targeted area.

8 APPENDIX I: Design Files

All documents and files required to run this project can be found on MakerRepo at the following link: https://makerepo.com/Daniyal/1667.personal-tracking-ecosystem-group-b