Simulation for efficient CO2 measurement system implementation in a realistic city environment

Panagiotis Grigoriadis
DTU, Autonomous Systems
s203269@student.dtu.dk

Dimosthenis Angelis
DTU, Autonomous Systems
s212893@student.dtu.dk

Kostantinos Spyrikos
DTU, Computer Science
s200240@student.dtu.dk

Bence Mány
DTU, Computer Science
s210278@student.dtu.dk

Nicolo Sponziello
DTU, Computer Science
s210226@student.dtu.dk

Vicente Bartual Ferran
DTU, Computer Science
s210110@student.dtu.dk

	Panagiotis	Dimosthenis	Konstantinos	Bence	Nicolo	Vicente
Environment	40%	40%	5%	15%	0%	0%
Sensor System	0%	0%	0%	0%	50%	50%
Code integration	50%	0%	0%	0%	0%	50%
User Interface	0%	0%	100%	0%	0%	0%
Evaluation	0%	30%	0%	70%	0%	0%
Results	0%	70%	0%	30%	0%	0%
Conclusion	0%	0%	50%	50%	0%	0%
Presentation	0%	0%	0%	80%	0%	20%

Table 1: Contribution table.

ABSTRACT

The measurement of CO_2 levels in a city has been a topic of broad research in recent years. There have been many approaches to solve this complex task. For this project, a 3D model of a city with its different components, such as buildings, trees and cars and the dynamics of the pollution, has been designed and simulated. Additionally, the CO2 level is measurement is also simulated by sensors that are placed all over the city. The goal of this project is to investigate through experiments which sensor implementation strategy should be followed in order to minimize the cost of the sensor system while maximizing the accuracy of their readings. This is done by tuning 3 parameters: the way the sensors are placed (static/dynamic), the distance between them and the sampling frequency.

Keywords

sensor, iot, network, city, simulation, CO_2, pollution

1 Introduction

As CO_2 levels have been worryingly rising over the last decades, we have reached a point where they pose a serious threat to people's health. This issue has been even more intense in major cities, where large numbers of people live. In this context, it becomes essential to have a dependable, effective system that measures the CO_2 within cities.

The purpose of this project is to create a simulation environment that will help to decide the optimal strategy to design the CO_2 sensor system. In order to achieve this, the simulation is based on a city with all the major elements that affect CO_2 generation and behaviour.

2 Problem

2.1 Problem Statement

What does the term "optimal strategy" mean in our case? It means that we need to implement a sensor system that is as cheap as possible, while simultaneously maintaining very high accuracy, measuring the city CO_2 level very close to the real value.

This of course comes with its challenges, as these two metrics, cost and accuracy, are inversely proportional. We could just fill our city with sensors, getting perfect accuracy, but the cost would be huge. On the other hand, we could add very few sensors and maintain a low cost, but it will also result in very inaccurate measurement.

At the end of the day, we need to experiment with different configurations, simulate them, compare their results, and select the best one according to our needs.

2.2 Initial Assumptions

We made the following assumptions in order to simplify the problem and minimize the computational load:

1. CO_2 cannot go into buildings.
2. CO_2 is only moved around by the wind, its natural diffusion and rain/snow/hail.
3. The environment is built up by 5m x 5m x 5m cells and the CO_2 concentration is constant inside them.
4. The environment is 180 x 180 x 3 cells large.
5. CO_2 leaving city boundaries is considered lost.
6. Only cars produce CO_2.
7. Cars are either gasoline or diesel fueled.
8. Trees consume CO_2.
9. Sensors have unlimited range of communication.
10. Sensor communication never fails (i.e. there is no packet loss).
11. Sensors only consume power when measuring and when sending the data.

3 System

3.1 System Diagram

Block Diagram The block diagram in Figure 3 represents the structure of our model[1]. The environment interacts with the city to influence the CO_2 levels (pollution), while the sensor system measures it (measured pollution), so that we can later compare them and draw conclusions.

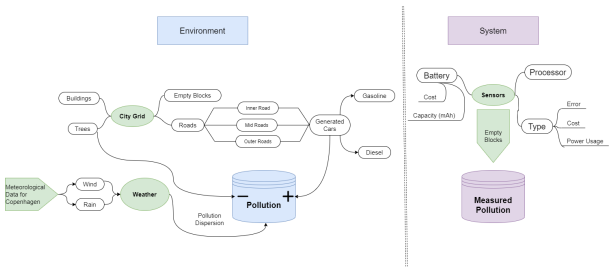

Figure 1: An abstract representation of the system.

City 2D Model Our city model was created online at `https://www.pixilart.com/`.

We created a 2D picture with pixels of different colors, each one representing a different entity that exists in our model. Consequently, we read this file with python, converting pixels, depending on their color value, to the corresponding python object (road/tree/empty/building). This created a 2D list that represents the ground layer of our city.

City 3D Model We converted our 2D city model to a 3D model by stacking the same the ground layer that we created in 2D 3 times, one on top of the other, only changing roads to empty blocks of air, for the 2nd and 3rd city layer. This created a 3D list that represents the entire city grid in 3D.

[1] In the figure, as "Empty Blocks" we describe the city blocks that don't contain a "building" or a "tree".

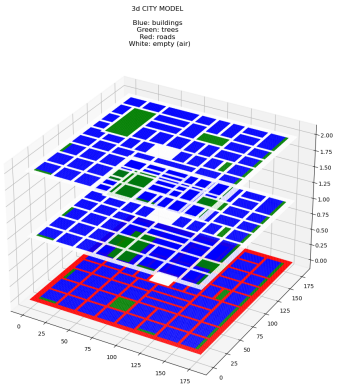

Figure 2: The 3d model of the simulated city.

As an assumption, our city is of size (180 row cells)x(180 column cells)x(3 city layers) and each cell represents a $5m^3$ cubic cell.

3.2 Environment

Cars The cars are the only source of CO_2 in our city. There are two types of cars, gasoline and diesel. The CO_2 that a car produces per iteration is equal to:

$$\begin{aligned} CO_2 generated = {} & car\ consumption * \\ & duration * \\ & speed * \\ & fuel\ CO_2\ factor \end{aligned}$$

- $CO_2 generated$ [grams]: The amount of CO_2 generated by the investigated car.
- $Car\ consumption$ [L/100km]: The fuel consumption rate of the car.
- $Duration$ [sec]: The duration over which we measure the generated CO_2.
- $Speed$ [km/sec]: The speed of the car.
- $Fuel\ CO_2\ factor$ [grams/L]: A CO_2 factor, which depends on the type of fuel used by the car.

The CO_2 factors are 33.64093867 (gr CO_2/L) and 38.5354738 (gr CO_2/L) for gasoline and diesel cars respectively.[SMH14]9

The positions of the cars in the city are dynamic. Every six hours we delete the previous cars and produce new ones based on the time of the day and the city zone in which the car is spawned. There is a 50% chance that a car is gasoline or diesel.

The number of cars changes every 6 hours. Depending on the six hour window of the day we are simulating, we have 10%, 100%, 100%, or 50% of the total 5000 cars going around the city.

Diffusion Diffusion is a natural process in which the transport of molecules is driven by concentration difference. This means that gas molecules move from a region of high concentration to a region of low concentration, which results in CO_2 naturally spreading and equalizing in the 3-dimensional space over time. The speed of this transfer also depends on the substance, but normally it is quite a slow process.

The exact physical equation is described by *Fick's first law* [Fic95]:

$$J = -D\frac{d\varphi}{dx}$$

- J [g/m^{2}s]: The diffusion flux, meaning the transported amount of substance per unit time per area of the delimiter medium (25 m^2 in our system).
- D [m^2/s]: The diffusion rate of the given substance, which is 1.6×10^{-5} m^2/s in the case of CO_2.
- $d\varphi$ [g/m^3]: The concentration difference between adjacent cells.
- dx [m]: the distance between the adjacent cells, 5 m in our system.

Using this equation, we can easily apply the diffusion effect in our city environment by iterating through every cell and calculating the substance flow between the current and the next cell. This will lead to a constant dispersion. However it is worth noting that the effect of diffusion is much slower than the wind, tree and rain effects, which means that we will never get a perfect uniform CO_2 distribution in our city.

Wind Wind is the most important factor affecting the CO_2 concentration in an area [Al-+20]. In our case, we simulated the meteorological conditions of Copenhagen [met].

This means that every day within a month that there is some sort of wind ($wind\ speed > 0$), the co2 will be moved to an empty adjacent cell (empty means that it doesn't contain a building or a tree) according to the wind direction and speed. Wind direction, in combination with the adjacent city entities (tree, building, empty, road) to the investigated cell will determine the direction, towards which CO_2 will move, while its speed determines how far away it will move.

Rain By assuming a heavy rainfall intensity and 100% CO_2 saturation of the water, we have defined rain as a meteorological phenomenon that makes the CO_2 drop to the bottom level of the city instantaneously.

This takes effect in regular intervals, that for the purpose of this simulation were based on meteorological data for the city of Copenhagen [met].

Trees An average mature tree absorbs 21 kg of CO_2 per year [ten]. This means that in any given day, a tree absorbs on average $21/365 = 0.06$ kg of CO_2. In our case, because we are interested in a city environment, we can safely assume that not all trees are fully grown. In this sense, we assumed that every tree absorbs 15kg of CO_2 per year.

This means that every day, a tree absorbs a specific amount of CO_2 from its adjacent cells, determined by the following formula:

$$daily\ trees\ CO_2\ absorption = \frac{15}{365 * \#\ of\ adjacent\ cells}[grams]$$

3.3 Sensors

While the environment generates and simulates the CO_2 level along the days, the IoT sensor system is responsible for taking measurements and collecting the data. This data will be compared to the "real" data produced to the enviroment to determine the system accuracy and thus make decision about the real system implementation.

The IoT sensor system consists several devices placed across the city. Each device is composed by different parts: a CO_2 sensor, a processor, a network module and a battery. Any combination of all these parts can create devices with different performance and characteristics. In this project, we have chosen the sCD4 CO_2 sensor and the ESP32 processor. As the network technology, we selected LORA, a low power protocol with a large range, but a relatively low data rate, perfect for the characteristics of our system.

For our project, we placed the devices only along the roads, since the cars are the only source of the CO_2 pollution.

Placement The devices are placed in the city at the beginning of the simulation. The density of the devices can be controlled by specifying the sensor distance parameter (if the distance is higher, there will be more space between each device and therefore the density will be lower, if the distance is lower, e.g. sensor distance = 1, the devices will be placed closer to each other therefore the density and the number of sensors will be higher). The number of devices has the most important impact on the overall accuracy of the system.

Moreover, the devices can change their position during the simulation. In particular, the placement strategy can be either STATIC or DYNAMIC. If the devices have a STATIC placement, their position won't change during the simulation and will stay in the same position they were placed initially. On the other side, if the devices have a DYNAMIC placement, their position changes over time. This could be achieved attaching the devices to, for example, the city buses.

The dynamic placement allows the devices to cover different areas of the city, but it can happen that they are too dense in some areas and more sparse in others, leading to a non uniform coverage of the city. Overall, the effect of the placement strategy greatly affects the accuracy of the system.

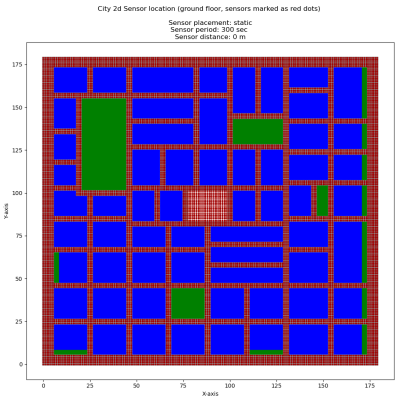

Figure 3: Sensors placed in every single block of the city.

Period In the real world, each device that is deployed on the environment has its own timing at which it takes a measurement. It is unrealistic that all the sensors wake up, measure and go to sleep at the same time in a system like ours, in which there is no communication between sensors. To simulate this aspect, each device has it's own starting time that is randomly chosen when the device is placed.

Moreover, to save power, the devices don't measure continuously but they exit from the sleep state and take a measure after a certain amount of time. This timing is defined by the sampling period parameter.

The CO_2 changes over time, so there is also a time aspect to it that can affect the accuracy. Measuring more often produces better time accuracy, but increases the power consumption.

Error Realistically, a sensor cannot measure with a precision of 100%, so every time a sample is being measured, it has a certain difference from the real value. This is due to multiple factors, like the physical characteristics of the sensor and the environment conditions.

To simulate this aspect, each sensor applies a random error every time it is used to take a measurement. Different sensors have different precision values. The error is specified as a value provided by its manufacturer.

To reduce the error, each time the device wakes up, instead of taking just one measurement, it takes five and calculates the average, obtaining a value closer to reality.

Battery In the real world, each device of the IoT system is powered by a battery or by a direct link to a electricity source such as the public electricity infrastructure. In our system, we considered devices that are powered by a battery. We determined that the devices should at least function for a year with one battery.

In fact, the simulation provides us, at the very end, with the information about the power consumed by one device in one year of continuous usage as configured in the experiment. The power consumed by a device is influenced by its components and the amount of measures it takes (sampling period). This allows us to take better informed design deci-

sions about the system that we are trying to implement. If for example, a device consumes 10000 mAh / year by taking one measure every 60 minutes, to run the system for 2 years we will need to equip our devices with a 20000 mAh battery.

Cost In general, placing a lot of devices will produce a system that has higher accuracy, but also higher cost. The more devices that are used, the more devices that have to be paid for. This is the main parameter affecting the cost.

Additionally, the sampling period also plays a role here. As stated before, taking more measurements means using more energy. After each simulation, the overall consumed energy is calculated and the battery is selected accordingly. We choose the smallest possible battery that allows the device to function for at least a year. The greater capacity comes with a greater price, therefore it also influences the device's overall cost.

It is the relationship between these parameters that we ought to explore, experiment on and eventually optimize. The placement strategy (static or dynamic) does not affect the cost of the system. To summarize, the total cost of the system is determined by the number of implemented devices and the sampling period.

4 Software

4.1 Software Tools

Python Python was used as the preferred programming language because of the team's background, its ease to use and the fact that it has a wide selection of libraries to work with and it is also ideal for data visualizations in 2d and 3d.

Trello Trello was used in order to keep track of all the tasks that each member had to do. Each task, depending on its state, would go to 1 of 4 lists: "To Do", "Ongoing", "Review", "Done".

Github Github was used to implement version control on the project's code (`https://github.com/gitglob/MBSE`).

GUI "Tkinter", a Python framework which is included in all standard Python distributions, was used in order to create a graphical user interface. This offered more convenient customization of the simulation parameters.

4.2 Implementation

The simulation is implemented in a main-loop function, similar to the design of video game control loops. The main loop constantly updates the environment, takes measurements from the devices and finally calculates the system's accuracy. The most important coding aspects of the project are the following:

1. Every city entity (building/tree/road/empty cell) is represented as an object.
2. The city is a 3D list with every element containing a cell entity.
3. The entire simulation takes place inside a while-loop that lasts for the desired simulation time that the user inputs as a parameter when executing the program.
4. The different effects on the city CO_2 levels are implemented as python functions, which are called sequentially inside the main while-loop.
5. The sensors system is also represented as python objects.
6. The sensor measurements are also implemented as functions and are the last things to be called during every iteration inside the main while-loop.
7. The measurement values and the real CO_2 are collected in lists at every iteration and are being compared at the very end of the simulation.

5 Evaluation

There are many ways to evaluate a simulation. We can set different inputs and take several outputs into consideration. Our chosen main inputs are:

- Distance between sensors - which is inversely proportional to the number of implemented sensors.
- Sampling period of sensors.
- Sensor placing strategy - static/dynamic.
- Time to run the simulation (number of days).

As a result of these inputs and the environment effects, we receive the following outputs:

- Overall cost of the system.
- Number of sensors.
- Root-Mean-Square Error of the measurements.
- Accuracy of the measurements.

5.1 Real CO2 vs Measured CO2

Depending on these inputs and the environmental effects, the simulation will produce random behaviour which we can measure. The effectiveness of the sensor network is based on the difference between the measured and the real values. The closer we can get, the better the system is. To visualize this test, we can plot both the measured and the real samples along the simulation.

5.2 Evaluation Metrics

There are different methods to describe differences between time series, which are going to be discussed in this section section. The two most suitable metrics are Root-Mean-Square Error and Accuracy.
The Root-Mean-Square Error is calculated by the following equation:

$$RMSE = \sqrt{\frac{\sum_{t=1}^{T}(\hat{y}_t - y_t)^2}{T}}$$

The accuracy is calculated as the average of relative accuracy of each measurement:

$$Accuracy = \frac{\sum_{t=1}^{T}(1 - (\hat{y}_t - y_t)/y_t)}{T}$$

- $\hat{y}_t$ is the average measured CO_2 level in the city in the given moment 't' and is defined by averaging all the sensor values along the city. Therefore it represents the whole city and not only a certain location.
- y_t is the average CO_2 level in the city in the given moment 't'. It is defined by averaging the CO_2 values of all non-building cells.

To perform these operations, we need the same amount of samples for the real (environment) and the measured values. Therefore, we only save the real environment CO_2 levels

when we perform the sensor measurements (they have the exact same period, which is set as a simulation parameter). Then, we can compare each elements and calculate the error and the accuracy between the two time series. We have to note here that the RMSE and the accuracy results are not completely detached values, but we have to calculate them separately, since the RMSE is an absolute number, and it is not quite intuitive. On the other hand, accuracy gives us a percentage, which is very easy to understand and interpret.

The efficiency of the implemented system was evaluated by comparing the total cost, which is calculated by summing the initial deployment price and the recurring costs based on the sampling frequency, and the accuracy of its measurements.

5.3 Custom cost-efficiency evaluation metric

In order to determine the most cost-efficient simulation, we have defined a cost-efficient factor 'ce' that is calculated as:

$$ce = \frac{Cost}{Accuracy * 100}$$

This factor is calculated for all the different simulation results. The one with the lowest ce value is the most cost-efficient.

5.4 Experiment Plan

We defined values for each of the simulation parameters of our experiments.

1. Sensor placement:
   (a) Static placement
   (b) Dynamic placement
2. Sampling Period:
   (a) Low: 15 minutes
   (b) Mid: 1 hour
   (c) High: 4 hours
3. Sensor Distance:
   (a) Low: 8 cells
   (b) Mid: 15 cells
   (c) High: 30 cells

This means that in total the simulation was tested across 18 experiments. By running all these different configuration combinations we looked for the most accurate and the most cost-efficient solutions to out problem.

6 Results

After each successful simulation, we will have many outputs, as it was mentioned in the Evaluation section. These outputs are being collected in a CSV file (*"results.csv"*), to be used to evaluate the results at any point, without the need to rerun the entire set of experiments.

6.1 Cost vs Accuracy

We run the simulation with the different parameters, as they are proposed in the experiment plan, and compared them. The detailed results of our experiments can be seen in the appendix.

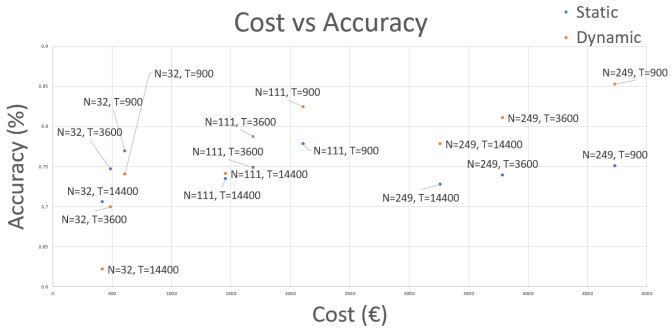

Figure 4: Plot that shows the cost vs the accuracy for all the simulations over one month. N and T stand for "number of sensors" and "sampling period" respectively.

After comparing them, we ended up with the following two solutions:

1. **The most accurate system**:
   (a) Number of sensors: 249
   (b) Sensor distance: 40 meters
   (c) Sampling time: 900 sec
   (d) Placement: Dynamic
   (e) Accuracy: 85.2%
   (f) Cost: 4731.0 €

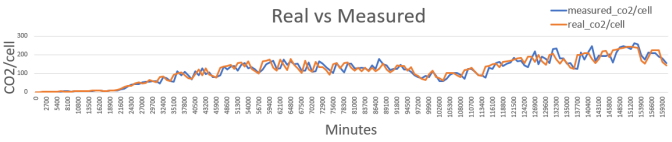

Figure 5: Measured vs real CO_2 per cell for the most accurate simulation over one month.

2. **The most cost-efficient system**:
   (a) Number of sensors: 32
   (b) Sensor distance: 150 meters
   (c) Sampling time: 14400 sec
   (d) Placement: Static
   (e) Accuracy: 70.6%
   (f) Cost: 419.2 €

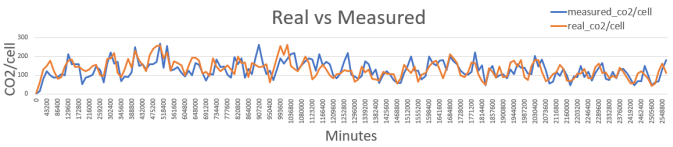

Figure 6: Measured vs real CO_2 per cell for the most cost-efficient simulation over one month.

The results can best be summarized by making a plot of "ce" over the different simulations.

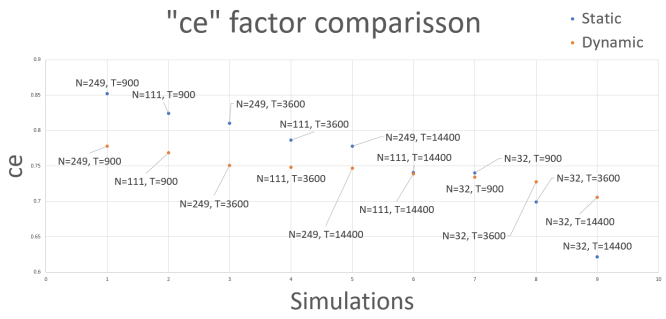

Figure 7: "ce" factor comparison for all simulations over one-month period. N, S and T stand for "number of sensors", "static placement" and "sampling period" respectively.

From figure 4, we can clearly see the effects of the different parameters: "number of sensors" and "sampling period" are definitely increasing the accuracy, also the price. On the other hand, "static/dynamic" sensor placement is more complicated: if we implement small amount of sensors, static placement results in higher accuracy, however with mid and high number of sensors, dynamic placement will have higher accuracy.

7 Conclusions

Determining the optimal strategy, in terms of price and accuracy, when it comes to CO_2 measurements for a city environment, is complicated. In our case, with a small, semi-realistic city environment, we managed to conclude that in order to achieve the highest accuracy, the best strategy to implement high amount of sensors with low sampling period, using dynamic placement. However, the most cost-efficient version might be more interesting for the costumers: implementing low number of devices with high sampling period and placing them statically. This way we can save the customer more than 4000€. At the end of the day,there is no "best" solution - it depends on the customers' needs. But with our simulation, it is easy and straightforward to visualize and select the optimal solution for different requirements.

However, we suggest that the reader takes our conclusions with a grain of salt. If our approach was to be followed by someone to implement a real sensor system in a real city, they would have to be way stricter, when it comes to the environment assumptions, and create a city model that is as close to the real one as possible. On top of that, it would be essential to have very accurate meteorological datasets for the investigated city. We also strongly suggest ask the advise of experts on several aspects of meteorology, and wind/particle dynamics and alter some of our python functions accordingly.

That said, we believe that our work isn't that far from reality and acts as a proof that this type of simulation would be a very helpful first step for a city that wants to implement a reliable CO_2 measurement system. This system can be used as a tool to evaluate its environmental footprint and improve the health quality of their citizens.

8 Difficulties And Future Work

8.1 Constraints

While creating a simulation, we can never reach the complexity of the real world, therefore we have to consider several simplifications, constraints in our model. Below there are the most crucial ones:

1. Our city model is a very simple, naive model and in order to extract more serious conclusions regarding the real world, one would have to use more advanced methods and create a much more realistic 3D environment.
2. None of our team members has the required expertise, when it comes to the topics of thermodynamics, CO_2 and meteorological phenomena, so some of our assumptions might not be entirely accurate, despite our best efforts in researching these topics.
3. The computational power of the computers that we own is limited, so we could not afford to apply the CO_2 generation and meteorological effects in every iteration of the main while-loop, as it is too costly. Even with the compromises that we made, simulating for periods longer that a couple of months was impossible.

8.2 Improvements

There are many ways to improve our current solution:

1. Functions can be optimized based on their individual computational needs.
2. Use different programming language that helps with performance (i.e. C++).
3. Use parallel programming instead of sequential.
4. Change some of the environment assumptions, so that the result is more realistic. For example:
   - Trees and buildings could be of various different sizes and shapes.
   - Buildings could actually contain some CO_2, as there are windows and doors through which CO_2 can move, which isn't taken into consideration so far.
   - The effect of rain could be more gradual, based on relevant scientific research.
   - Wind effect could become more realistic, based on relevant scientific research.
5. Add some parameters, when it comes to the sensor system. For example:
   - Add sensor failure chance (%).
   - Add sensor lifespan.
   - Add battery lifespan.
6. Include in the simulation the packets sent by the devices of the IoT system using the network. This was not implemented in our simulation since it does not offer any useful insight in order to find an answer to our question. In the real world, the communication between the devices plays a peculiar part in an IoT system, especially when different technologies or protocols comes in to play.
7. Perform a mathematical analysis on the "results.csv" file, in order to understand which one out of the simulation parameters affects the system the most, and then focus on understanding and fine-tuning that. Some models that could be investigated are simple Linear

Regression models and Principal Component Analysis. However, in that case, many more simulations should be ran, so as to gather a significantly larger dataset.

9 Acknowledgments

We would like to thank Edward Alexandru Todirica (Associate Professor at DTU Compute, Department of Applied Mathematics and Computer Science at the Technical University of Denmark, DTU) and Paul Pop (professor at DTU Compute, Department of Applied Mathematics and Computer Science at the Technical University of Denmark, DTU) for their precious comments on our project. Their guidance was crucial in order to coordinate and implement all the different aspects of this project.

References

[Fic95] Adolph Fick. "On liquid diffusion". In: *Journal of Membrane Science* 100.1 (1995). The early history of membrane science selected papers celebrating vol. 100, pp. 33–38. ISSN: 0376-7388. DOI: https://doi.org/10.1016/0376-7388(94)00230-V. URL: https://www.sciencedirect.com/science/article/pii/037673889400230V.

[SMH14] Natalia Sobrino, Andrés Monzón, and Sara Hernandez. "Reduced Carbon and Energy Footprint in Highway Operations: The Highway Energy Assessment (HERA) Methodology". In: *Netw Spat Econ* 16 (Apr. 2014), pp. 1–20. DOI: 10.1007/s11067-014-9225-y.

[Al-+20] Russell M. Al-Bayati et al. "The relationship between the concentration of carbon dioxide and wind using GIS". In: *AIP Conference Proceedings* 2290.1 (2020), p. 050042. DOI: 10.1063/5.0027402. eprint: https://aip.scitation.org/doi/pdf/10.1063/5.0027402. URL: https://aip.scitation.org/doi/abs/10.1063/5.0027402.

[met] meteoblue. *Simulated historical climate weather data for Copenhagen*. URL: https://www.meteoblue.com/en/weather/historyclimate/climatemodelled/copenhagen_denmark_2618425.

[ten] tenmilliontrees. *All about trees*. URL: http://www.tenmilliontrees.org/trees/.

APPENDIX

A Appendix A

Our appendix contains a number of different pictures, which illustrate the effect of the different python functions that we have implemented.
In every picture, it is worth pointing out that the geographical orientation relationship with the coordinate system is the following:

- North -> +Y
- South -> -Y
- East -> +X
- West -> -X

A.1 System

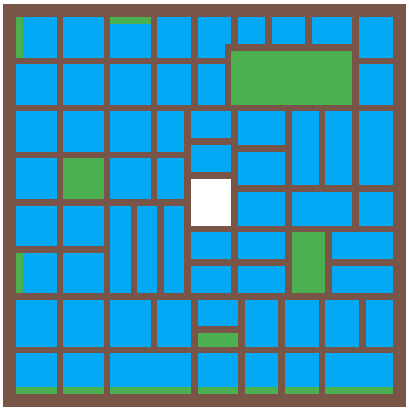

Figure 8: The ground layer of our city model, as drawn in https://www.pixilart.com/. The color "white" represents the empty blocks (the city square), the color "green" represents the trees, the color "blue" represents the buildings and the color "brown" represents the roads.

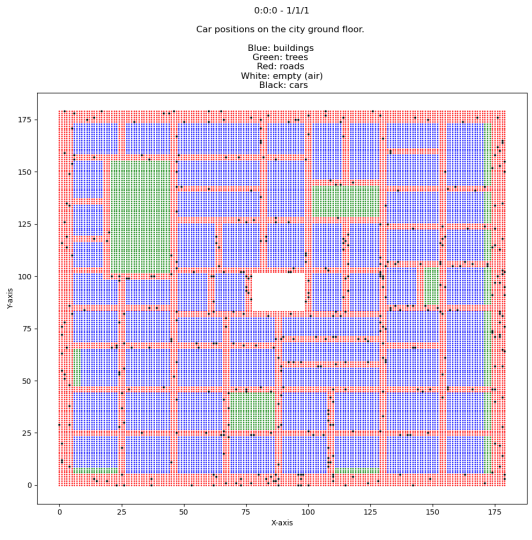

Figure 9: An example of the randomly spawned positions of cars.

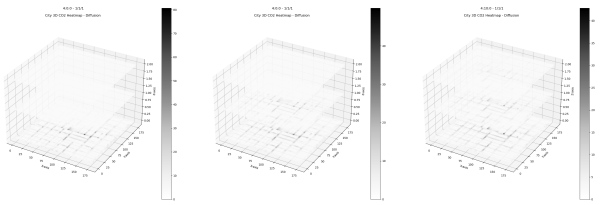

Figure 10: The effect of diffusion within two iterations.

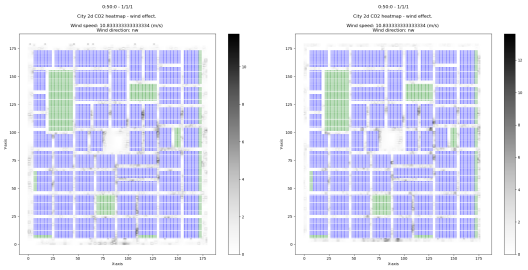

Figure 11: The effect of wind within one iteration. The wind directions with respect to the coordinate system are as following:

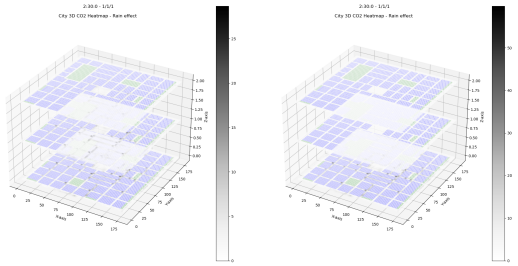

Figure 12: The effect of rain within one iteration.

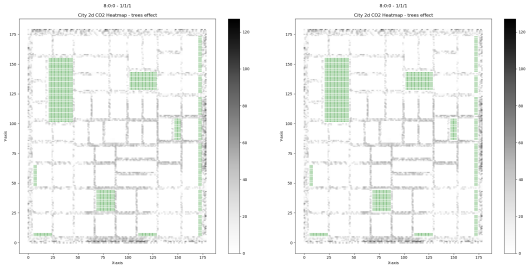

Figure 13: The effect of trees within one iteration, but significantly augmented so that it is more visible to the viewer.

A.2 Results

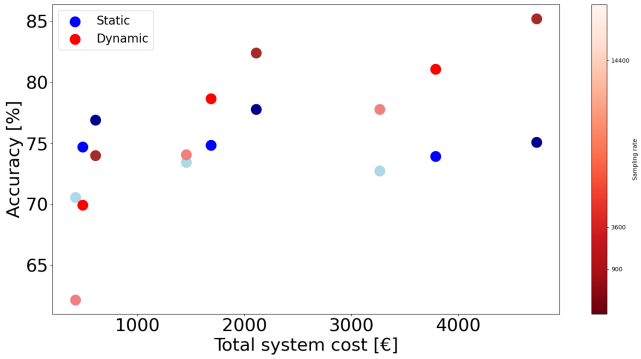

Figure 14: Experiment results plotted. The darker the colour, the higher the sampling frequency.

	A	B	C	D	E	F	G	H
1	Cost	Sensor dis	Number o	Static	Sampling	Simulation time	Error	Accuracy
2	419.2	30	32	FALSE	14400	2592000	0.5041	62.1637
3	608	30	32	TRUE	900	2592000	0.2979	76.9003
4	2109	15	111	TRUE	900	2592000	0.2915	77.8025
5	419.2	30	32	TRUE	14400	2592000	0.3455	70.5604
6	486.4	30	32	TRUE	3600	2592000	0.3213	74.6959
7	486.4	30	32	FALSE	3600	2592000	0.3852	69.9349
8	3261.9	8	249	FALSE	14400	2592000	0.2647	77.8029
9	1687.2	15	111	TRUE	3600	2592000	0.3114	74.8512
10	3261.9	8	249	TRUE	14400	2592000	0.285	72.7534
11	3784.8	8	249	TRUE	3600	2592000	0.2857	73.9206
12	1687.2	15	111	FALSE	3600	2592000	0.2626	78.6807
13	608	30	32	FALSE	900	2592000	0.3601	74.0129
14	3784.8	8	249	FALSE	3600	2592000	0.2337	81.0726
15	1454.1	15	111	FALSE	14400	2592000	0.2932	74.0808
16	2109	15	111	FALSE	900	2592000	0.2228	82.4201
17	4731	8	249	TRUE	900	2592000	0.2985	75.0839
18	4731	8	249	FALSE	900	2592000	0.2017	85.2318
19	1454.1	15	111	TRUE	14400	2592000	0.2829	73.4373

Figure 15: Experiment results for one month of simulation.

B Project Management

B.1 Scrum Roles

The scrum roles, that are shown below in Figure 16 were chosen in order to make the process more effective and structured for everyone's benefit. More precisely:

- Product Owner: For the product on which they're working, this role represents the customer and the company as a whole. Before each sprint, they take ownership of the backlog and work to prioritize issues to be worked on. On a daily basis, they make executive product choices. In the end, they're converting client requirements into work items for the Development team.
- Scrum Master: This individual is in charge of ensuring that the team has everything they require to deliver value. They schedule meetings and keep track of progress and roadblocks. This role represent everything that a project manager should be doing, but filtered via the scrum lens.
- Development Team: This is a group of people from several departments that are all dedicated on delivering a working software. It is the singular noun for all developers, designers, QA testers, and other technical positions involved in the actual product development.

This group of 5-9 persons should ideally be committed to just one scrum team. It could appear a little different in reality. With effective facilitation by the Scrum Master and Product Owner, the development team may be self-organizing and driven to produce value.

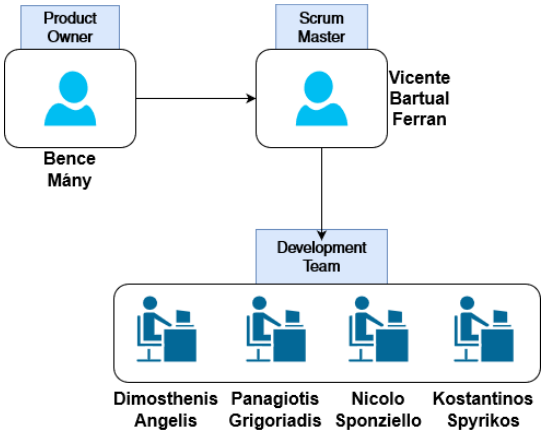

Figure 16: Scrum Roles

B.2 Trello

The collaboration tool Trello (Figure 17) was chosen in order to organize this project into boards. With Trello, everyone can have an idea on what the current topic of focus is at the moment, what is being implemented in the specific period of time and what is expected to be delivered in a certain deadline and by whom.

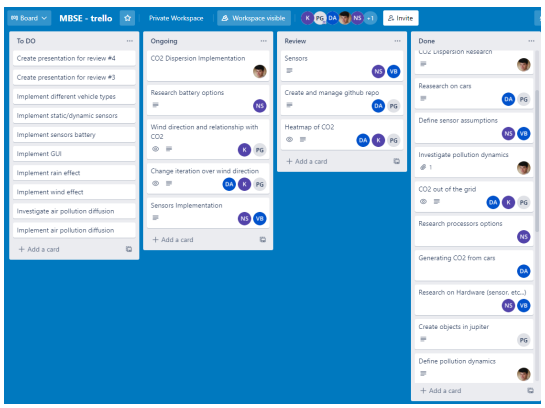

Figure 17: We used Trello in order to manage our project needs and tasks.

In Figure 17, a screenshot of this project's Trello board is presented and there is a clear overview of the different boards:

- To Do
- Ongoing
- Review
- Done

C How To Run

In order for someone to run our simulation tool, he must follow the next steps:

1. Clone the GitHub repository.
2. Open a new terminal, change your directory to the project folder and type:
   pip install -r requirements.txt, in order to install all the required dependencies.
3. To run the simulation: *python -m main*
4. The graphical user interface (Figure 18) will appear.
   - Press *START the Simulation*, if you want to execute the simulation with the default values.
   - Press *Set Parameters*, in order to modify the parameters of the simulation.
     · Remember to always click on the corresponding button of a parameter, after you set its desired value.
     · When you are done, just press *THE PARAMETERS ARE SET !!* and then you are ready to start the simulation with the desired parameters.
   - Press *'Run Multiple Simulations'*, in order to run our experiment plan. You can always modify the experiment plan in 'main.py -> lines 28-30'.
     · In the new window that appears you can set the days of the simulation.
     · Remember to always click on the corresponding button of a parameter, after you set its desired value.
     THE PARAMETERS ARE SET !! and then you are ready to start the simulation with the desired parameters.
   - Press *QUIT* if you want to exit the simulation and close all the windows.

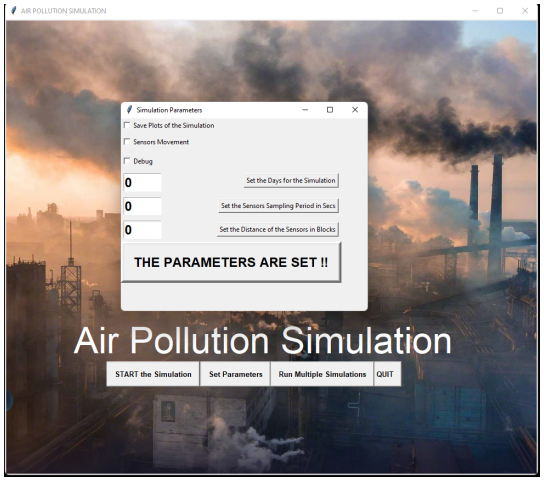

Figure 18: The simulation graphical user interface (GUI).