Sensible Meeter

Wildan Zulfikar, Anh-Duc Vu, Robert Pakkanen, Olli Bisi

Abstract

Amount of co2 in indoor air causes decrease in productivity and health. This issue is relevant to for example students and office workers. We have created an application which measures and predicts the co2 level so that the user can mitigate these effects.

1. Introduction

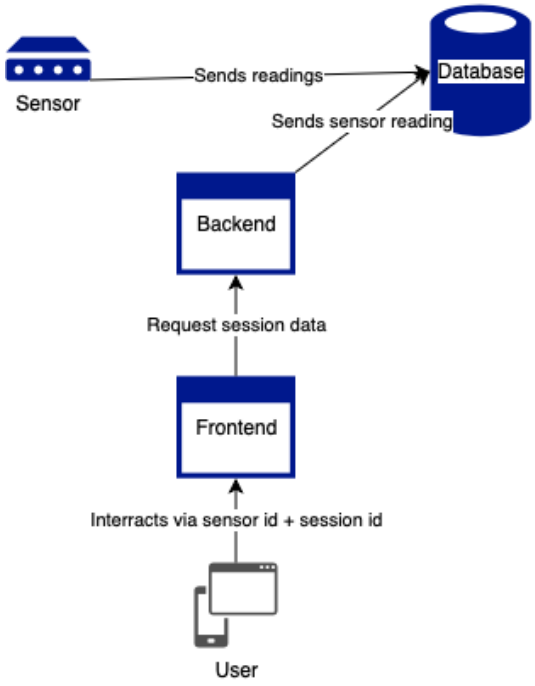
A high co2 concentration causes for example tiredness and headaches, which can lead to low productivity. As amount of co2 slowly creeps up, it usually can take some time for people to realise that they should take a break. At this point tiredness or other symptoms have already set in. This could have been avoided by taking a break and getting fresh air before exposing oneself to higher concentrations for periods of time. While the outlying issue, bad ventilation, is a harder problem to fix, for individual taking a break can improve productivity and health with easy solution: taking a break. Taking a break of course is easy to do (mostly) and every one knows this, but it is too easy to get stuck at working. An 'outside' prompt might provide the required push to action. This is why the application was created. Our application monitors the level of co2 in room, and predicts the value of co2 for the next ten minutes. If the predicted timeframe is above a safe threshold, user is prompted. Our application is a way of reminding to take a break.

2. Goals

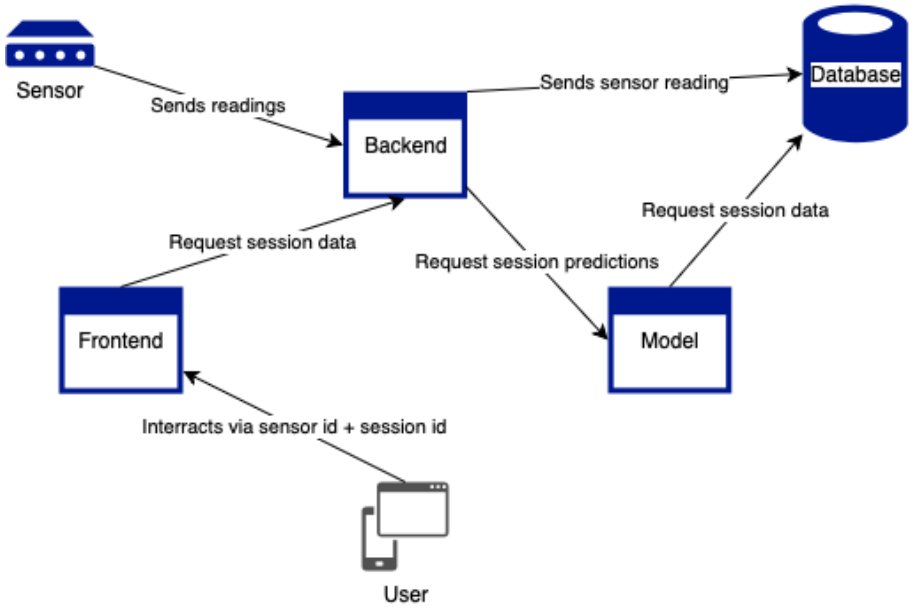
The goal of this project was to create a web-application which would utilise our co2 sensor. The application allows users to see the measurements of their sensor in near real-time to see the co2 levels in the room they are in. The application also predicts co2 levels for the next 10 minutes. As the predicted co2 level starts to be too high, user is notified and prompted to take a break.

3. System Architecture and Design

Our systems consists of raspberry pi connected to a co2 sensor and a web application. As we only did a proof-of-concept, we simplified the goal architecture a bit:



Original idea was to host the application online to allow multiple sensors. This would've required the application logic to be a bit different, as we would need the sensors to send their readings to backend instead of straight to database. We also hostel the model directly in the backend, unlike the we planned originally. Original architecture plan below:



3.1. Hardware

Component/Device	**Use**
Raspberry pi 3b+	We used this device for controlling and sending measurement
DFRobot co2 sensor v1.2	This sensor was used to measure co2 readings

We used the co2 sensor by integrating it with raspberry pi. The raspberry pi then sent the readings to our database which was running in cloud.

3.2. Software

Component	Use
Sensor	Node script that reads sensor readings and sends them to backend
Backend	Python Flask application that runs in docker
Model api	Python Flask application that runs in docker
Frontend	Node script that runs in docker
Postgres	Database hosted in metabase
Model	LSTM model for co2 level prediction

The goal architecture was to run application in three parts: frontend, backend and model. User would interact with frontend. Frontend would interact with backend. Backend would interact with sensor, database and model.

4. Addressing Challenges

First challenge was to figure out what we are doing. We want to measure co2, and give predictions based on that. How do we measure? We got readings from the sensor, and we decided to post them once per second to database.

We immediately realised that our sensor gave weird readings. Instead of a value that would grow, we had a high starting value which then decreased over time. We found out that sensor actually reads voltage which needs to be converted to co2.

Once we got our sensor working and sending readings in correct format, we needed to figure out which readings to use. We needed to start planning the application architecture. However we ran into some issues with our database provider and local python environments, which slowed our development time down drastically. This is why we decided to focus on getting an application to run locally and simplified the architecture.

For model the first issue was on what to predict. We decided that we will do a time series model that predict next ten minutes of co2 levels. This way the user will have a choice to start making preparations for ending meeting, but not prompted with ”stop what you are doing and get out right now”, which, of course, is unlikely to happen, and prompt is ignored. A safer ”start preparing to take a break now” allows user to ease out of current work.

Second issue for model were which features to use. We decided to use only co2 reading with timestamp. This does not give a full picture of co2 accumulation, as there are multiple things that contribute to it: number of people in the room, size of the room, ventilation of the room to say the least. However, to be able to apply these to our model, we would require training data for each of the scenarios, and a way to keep track of different rooms, ventilation level and number of people. Also, from the ease of use perspective, the less user input is required the better. As we do not have data of different room sizes, ventilation levels or the number of people in rooms, we would need to ask them from the user. Who would then need to estimate them, which could also be a horribly wrong estimation. Our intuition for prediction is that the rate of accumulation is the important feature here, not why it is increasing at such a rate. So, we are assuming that the rate of co2 accumulates in a similar fashion with two people versus one people in a

larger room. This might be a erroneous hypothesis, but for this project we lack timetable and resources to do a more through research on the topic.

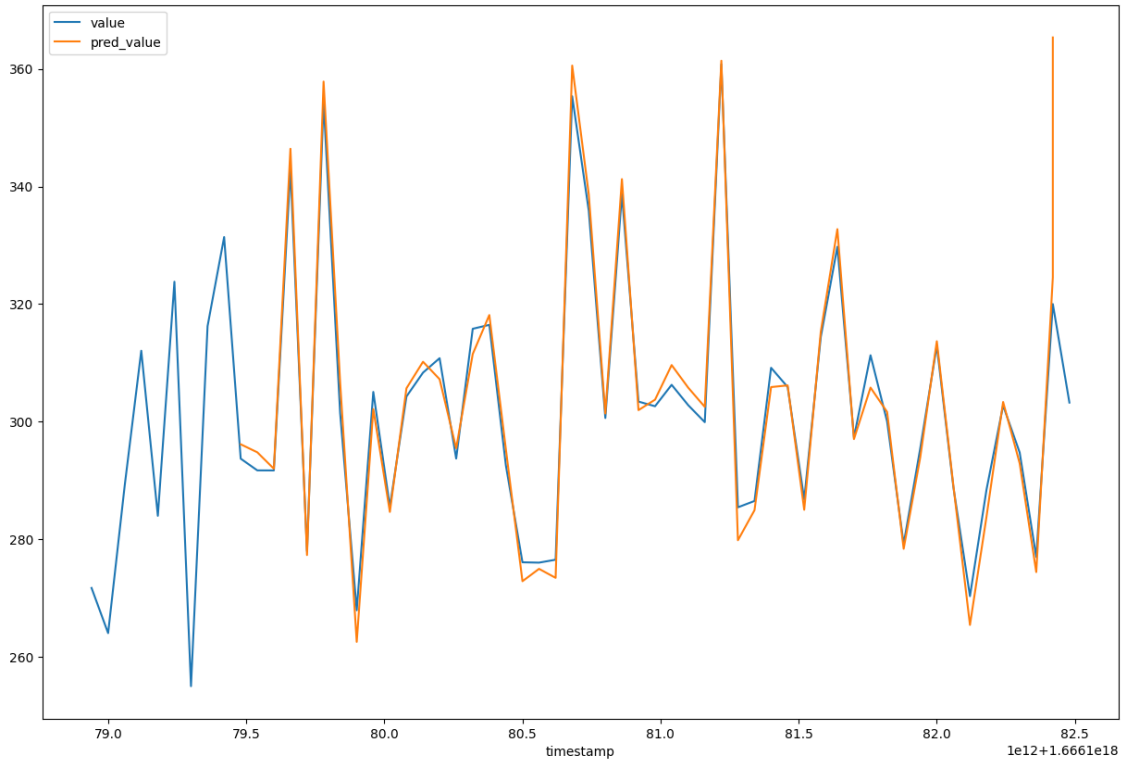
As the sensor saves readings once per second, we decided to take an average of the readings per minute, and use the aggregated value as input for the model.

5. Performance Evaluation and Testing Results

For the model evaluation we created a ”baseline” model which we compared models to. The baseline was taking average of sequential measurements for ten minutes, and using that for predicting next ten minutes by adding the averaged difference to last measurement ten times.

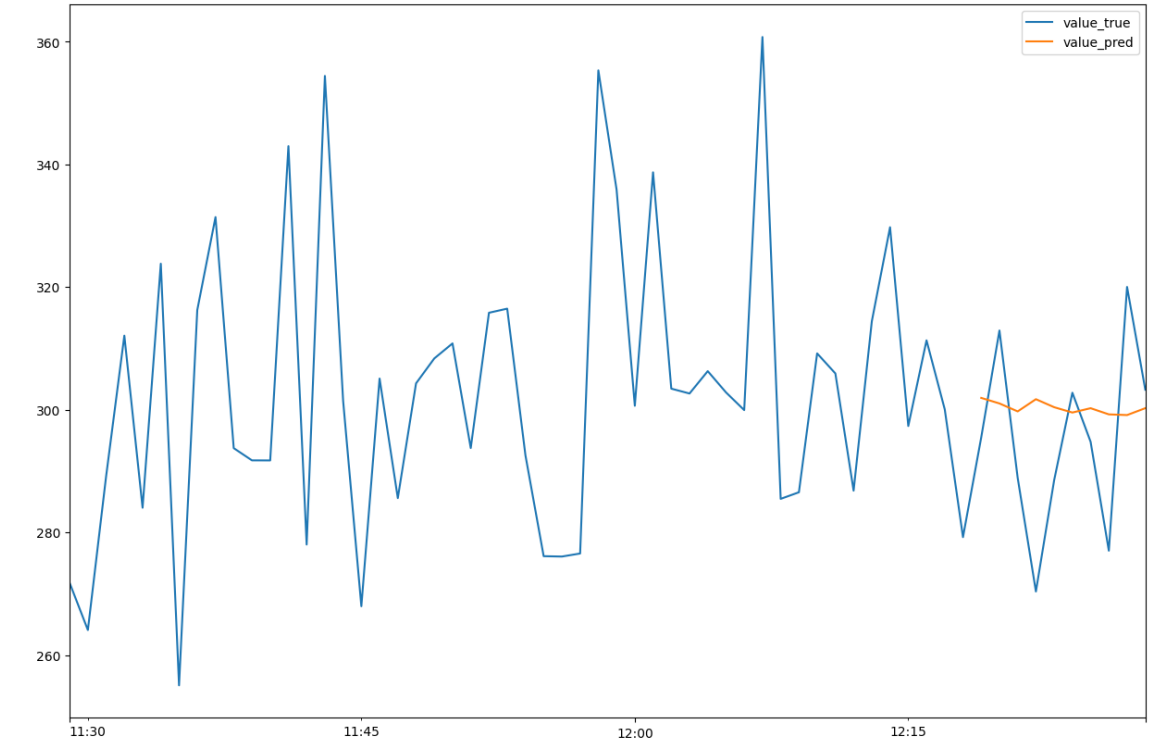
We did a split of training data which was generated by the sensor in various rooms with varying amount of people. This data was then split to train and test set. The results are were then compared.

For the baseline model, the performance was quite poor. As the model reads the last 10 readings, any spike might be amplified, as can be visually seen here:



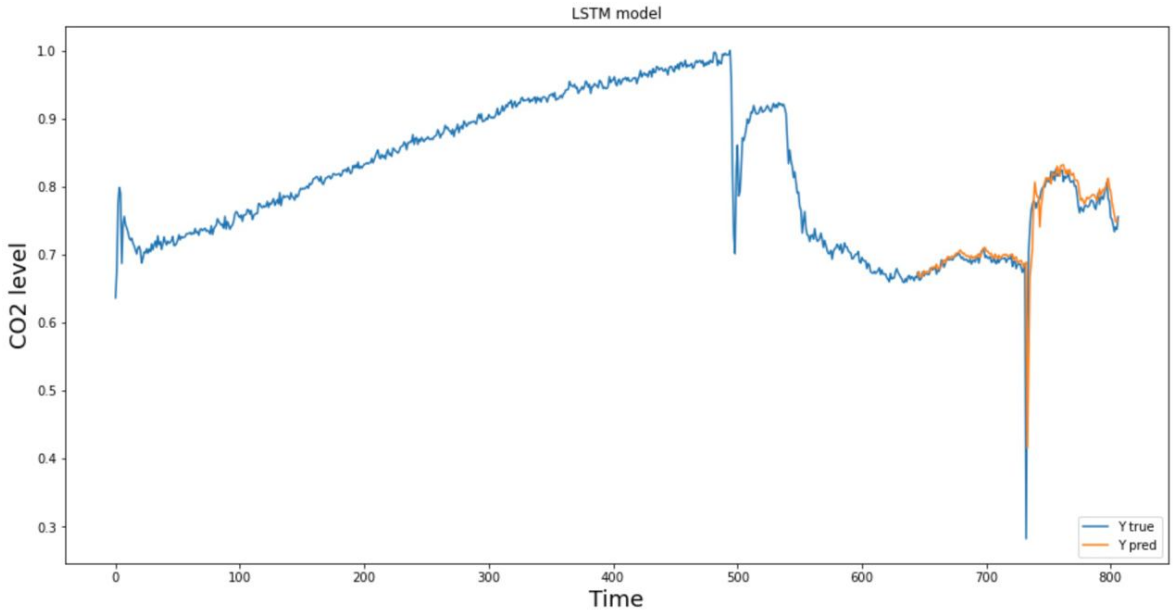
Better model would likely be to take all readings seen, but we wanted to keep the same input - output ratio as in other models.

Additionally we tried an autoregressive model, where predictions were done with the same amount of previous readings:



This model did better than the baseline model, but still not great.

Last model that was tested was Long-short term memory neural network (LSTM):



As can be seen, the results are very good. This is why we decided to use LSTM model in our application.

6. Concluding Remarks and Avenues for Future Work

This project was mostly a proof of concept. We tried to mitigate the effect of co2 indoors for people. The issue we tried to solve however is a real one, and there is no quick fixes to it. Well, other than taking a break. Better indoor air quality is something everyone hopes, but can be expensive to implement. A need for breaks while working is known, and our system could provide a push to actually do it. The actual device should run on battery to be actually usable. Current way of plugging in is not optimal, and would hinder real life applications. Best solution would be either a battery powered small box that would have on/off button, or a device that would connect to a phone or laptop via cable.

This project could be taken to a different direction altogether: the sensors could be added to rooms themselves, and then users could be prompted. This would of course need a more thorough design and would be a lot more expensive. This would allow for building owners to monitor room usage and air quality in specific rooms. This would of course allow more features for the model, but would be harder to maintain. Such systems also most likely already exist.

7. Availability

The repository can be found from GitHub: https://github.com/obisi/sensible-meeting

Our demo video can be found from google drive: https://drive.google.com/file/d/1SnDjUoAhJMjkA0IMq2nUH5A4vcb-E4FX/view?usp=share_link