Markus-Philipp Gherman, BSc

Towards Intelligent Air Quality Sensing on Intermittent Power

MASTER'S THESIS

to achieve the university degree of

Master of Science (Diplom-Ingenieur)

Master's degree programme: Information and Computer Engineering

submitted to

Graz University of Technology

Supervisor

Olga Saukh, bak. Assoc.Prof. Dr.rer.nat. MSc

Institute of Technical Informatics

Graz, February 2021

AFFIDAVIT

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master's thesis.

Date, Signature

Abstract

Popular low-cost air quality sensors embedded into IoT devices are based on metal oxides (MOX) that change their electrical resistance in response to ambient pollutants emitted as gases. Continuous MOX sensor operation requires heating up the sensor's hotplate to several hundred degrees and thus induces high energy cost. A common way to save energy is duty cycling the sensor. However, chemical reactions on the sensor's surface take time and fail to run to completion as the duty cycle shortens. As a result, sensor sensitivity to various gases deviates from a continuously operated sensor. In this thesis, we show that it is possible to recover an accurate continuous sensor measurement from transient responses obtained from a duty cycled sensor operated with small on-times by Machine Learning Models. We achieve a mean absolute error (MAE) of 143*ppb* for tVOC and 80% of indoor air quality levels correctly predicted. Our models are invariant to minor baseline shifts and work for both tVOC and CO2-eq signals provided by the sensor. By applying duty cycling and ML Models, we can save 98% of the sensor's energy consumption and still provide accurate measurements. An additional difficulty arises if the sensor is operated on-demand due to intermittent energy availability. We propose a compensation algorithm for this issue by mapping on-demand measurements to virtually duty cycled readings in the value domain and discuss how to further reduce the error of on-demand sampling by a sensing-friendly scheduling in case the sensor is to be operated on intermittent power.

Kurzfassung

Preiswerte Sensoren zur Luftqualitätsmessung, welche vor allem in Internet of Things-Anwendungen eingesetzt werden, basieren auf Metalloxiden (MOX), deren elektrischer Widerstand sich in Abhängigkeit von Luftverunreinigungen ändert. Der kontinuierliche Betrieb eines solchen Sensors erfordert die Erhitzung einer Mikrokochplatte auf einige hundert Grad, was mit einem hohen Energiebedarf verbunden ist. Eine verbreitete Methode, den Energiebedarf zu senken, ist *Duty Cycling*, in welcher der Sensor für eine bestimmte Zeit eingeschalten und wieder ausgeschalten wird. Chemische Reaktionen innerhalb des Sensors benötigen jedoch Zeit und können deshalb je nach Länge der Einschaltzeit nicht mehr vollständig ablaufen. Die Empfindlichkeit eines Sensors, welcher auf diese Art betrieben wird, weicht daher von der eines Sensors, welcher kontinuierlich betrieben wird, ab. In dieser Arbeit zeigen wir, dass man basierend auf den Messungen in den ersten Sekunden des Einschaltvorganges eines in *Duty Cycle* mit kleinen Einschaltzeiten betriebenen Sensors mit der Hilfe von Modellen basierend auf maschinellem Lernen (ML), Messungen, wie sie ein kontinuierlich betriebener Sensor liefert, erhalten kann. Wir erreichen hierbei einen Mittleren Absoluten Fehler (MAE) von 143*ppb* für tVOC und schaffen es, 80% der Innenraumluftqualitätsstufen richtig zu berechnen. Unsere Modelle sind robust gegen kleine Änderungen der Grundlinie der Sensoren und funktionieren für die Sensor Signale tVOC and CO2-eq. Durch die Anwendung von *Duty Cycling* und ML Modellen, können 98% des Energiebedarfs des Sensors eingespart und trotzdem zufriedenstellende Messungen bereitgestellt werden. Wird der Sensor jedoch nicht in regelmäßigen Abständen ein- und ausgeschalten, sondern aufgrund unterbrochener Energieverfügbarkeit unregelmäßig betrieben, wie zum Beispiel mit Solarenergie, so treten weitere Schwierigkeiten auf. Hierfür stellen wir einen Korrektur Algorithmus vor, mit welchem unregelmäßige Messungen in virtuelle regelmäßige Messungen umgewandelt werden können und diskutieren, wie die durch unregelmäßigen Betrieb auftretenden Fehler durch eine optimale Vorbereitung der Messzeitpunkte verringert werden können.

Acknowledgements

My journey through university was an unforgettable experience, with many great people along its way.

I am very thankful for having my family, which has given me support at all times. Without you, many things would not have been possible.

Many obstacles on the way, which seemed hard to solve at first, were well solved in the end with the help of my friend Fabian. Our never ending discussions on various topics were all worth it, thank you!

I would also like to thank Olga for guiding me through my thesis and always providing help, no matter the question or the time. I really appreciate your effort! Thank you very much!

Andrés and Yun, the discussions with you helped me to get new ideas and to try out new things, thanks a lot!

And my greatest gratitude goes to you, Ina. Thank you for your patience, kindness, encouragement, care and love through all the ups and downs!

Graz, February 2021 Markus-Philipp Gherman

Contents

1 Introduction **19**

2 Related Work **21**
2.1 Environmental sensors . . . 21
2.2 Low-power measurements improved with ML Models . . . 22

3 Hardware **23**
3.1 SGP30 Gas Sensor . . . 23
3.1.1 Target Gases . . . 24
3.2 Metal-Oxide Gas Sensing Principle (MOX) . . . 25
3.3 Sensor Platform . . . 26
3.3.1 Build . . . 26
3.3.2 Taking Measurements . . . 27
3.4 Operation Modes . . . 28
3.4.1 *Continuous Mode* . . . 29
3.4.2 *Duty Cycle Mode* . . . 29
3.4.3 *On Demand Mode* . . . 31
3.5 Energy Consumption . . . 32
3.6 Sensor Differences . . . 34
3.7 Measurement Units: From *ticks* To *ppb/ppm* . . . 35

4 Data Collection **37**
4.1 $D_{train+test}$ & D_{test}^{+4d} . . . 38
4.2 D_{test}^{+18d} . . . 40
4.3 D_{test}^{+22d} . . . 41
4.4 D_{test}^{+53d} . . . 42

5 Machine Learning Models & Data Preprocessing **45**
5.1 Machine Learning Models . . . 45
5.1.1 RNN . . . 45
5.1.2 LSTM . . . 46
5.1.3 GRU . . . 48
5.1.4 XGBoost . . . 49
5.1.5 AutoML . . . 49
5.2 Data Preprocessing . . . 50
5.3 Evaluation Metrics . . . 51

6 Experimental Work **53**
6.1 Predicting *Continuous Mode* Signal from *Duty Cycle Mode* Signal 53
6.2 *On Demand Mode* Analysis and Correction 64

7 Conclusion & Future Work **69**

Bibliography **71**

List of Tables

3.1 Calculated Energy Consumptions 33

4.1 Detailed information about the recorded datasets 38

6.1 P1. ML Models Performances on $D_{train+test}$ Test Set 54
6.2 P1. ML Models, $W160$, applied on future data 58
6.3 P1. GRU Model, $W160$, applied on future data with offsets 60

List of Figures

3.1 tVOC concentrations related to IAQ levels . . . 24
3.2 CO2 concentrations and details . . . 25
3.3 Visualization of the Metal-Oxide Gas Sensing Principle (MOX) . . . 25
3.4 Sensor Platform . . . 27
3.5 *Duty Cycle Mode* Transient Responses . . . 30
3.6 *On Demand Mode* Transient Responses . . . 31
3.7 Sensor Platform Unit Schematic . . . 32
3.8 SVM30 Sample Measurement using RocketLogger . . . 33
3.9 *Device 1* and *Device 2* operated in *Continuous Mode* . . . 34
3.10 *Device 1* and *Device 3* operated in *Continuous Mode* . . . 35
3.11 Alignment of tVOC and concentration c . . . 36

4.1 Datasets Timeline . . . 38
4.2 $D_{train+test}$ Plot . . . 39
4.3 D_{test}^{+4d} Plot . . . 40
4.4 D_{test}^{+18d} Plot . . . 41
4.5 D_{test}^{+22d} Plot . . . 42
4.6 D_{test}^{+53d} Plot . . . 43

5.1 RNN Structure Visualization . . . 46
5.2 LSTM Cell Visualization . . . 47
5.3 GRU Cell Visualization . . . 49

6.1 P1. GRU Model applied on $D_{train+test}$ Test Set in *ticks*, $W160$ . . . 55
6.2 P1. GRU Model applied on $D_{train+test}$ Test Set in *ppb*, $W160$ . . . 56
6.3 P1. AutoML Model applied on $D_{train+test}$ Test Set in *ppb*, $W160$ . . . 57
6.4 P1. GRU Model applied on D_{test}^{+4d} in *ppb*, $W160$ . . . 59
6.5 P1. GRU Model applied on D_{test}^{+22d} in *ppb*, $W160$ . . . 59
6.6 P1. GRU Model applied on D_{test}^{+53d} in *ppb*, $W160$ . . . 60
6.7 P1. Error Histogram over offsets for GRU applied on future data . . . 61
6.8 P1. GRU Model applied on D_{test}^{+22d} in *ppb*, $W160$, with offset . . . 62
6.9 P2. GRU Model applied on $D_{train+test}$ Test Set in *ppm*, $W160$ . . . 63
6.10 P2. GRU Model applied on D_{test}^{+4d} in *ppm*, $W160$ . . . 63
6.11 Visualization of *On Demand Mode* band . . . 65
6.12 Histograms showing errors before and after correction . . . 66
6.13 Histograms showing errors before and after correction + time average . . . 66
6.14 *On Demand Mode* correction on $D_{train+test}$, $W20$ . . . 67

Chapter 1

Introduction

Wirelessly connected low-cost sensors are the key enabling technology for intelligent Internet of Things (IoT) applications. However, numerous important classes of environmental sensors, such as chemical sensors, are too power hungry to serve battery-powered IoT applications.

Regularly duty cycling a sensor provides a considerable reduction of energy consumption, yet significantly changes sensor sensitivity and thus affects the accuracy of the measurement. The same happens for on-demand (irregular) operation due to intermittent power availability, where also additional difficulties arise. Therefore, most gas sensors are designed for continuous operation, coming with the cost of high energy consumptions.

The sensor used to conduct experiments in this thesis is the SGP30 [Sen20c] gas sensor from Sensirion used for measuring tVOC and CO2-eq, *e.g.*, in air purifiers or IoT applications. This sensor is recommended to be constantly powered, resulting in high energy consumption. To lower the energy consumption, we duty cycle this sensor and operate it on-demand with on-times of up to 5 seconds and analyze the occurring effects.

The research question we investigate in this work is: *Can we compensate for sensitivity differences due to a changed operation mode in order to reduce energy consumption?*

Challenges Predicting accurate measurements from a transient response, representing the values obtained from the sensor during its short on-time, sampled either regularly or irregularly, corresponds to predicting the outcome of a chemical reaction with minimum observation time. The prediction model should provide a reasonable compensation for the following three issues: (1) Sensor's sensitivity changes due to duty cycling. (2) The asymptotic settling value of the observed transient is very different from the measured values in the continuous mode, since chemical reactions do not run to completion and build byproducts on the sensitive surface. (3) MOX sensitivity changes over time depending on the purity of the operating environment and operation mode.

Contributions and road-map In this work, we give positive answer to the above research question and tackle all above mentioned challenges. We build predictive Machine Learning Models to recover the measurements of a continuously operated sensor from transient responses measured while duty cycling with on-times below 5 seconds, obtaining a MAE of 143ppb between ground truth measurements and model predictions for tVOC and predicting 80% of indoor air quality levels correctly, and thus showing feasibility for the usage of ML Models for duty cycled gas sensors. We further investigate the properties of transients obtained from a sensor operated on-demand with on-times below 5 seconds,

propose a correction algorithm for unwanted effects induced by on-demand operation and how to further reduce errors with a sensing-friendly scheduling on intermittent power.

In this thesis, Chapter 2 summarizes related work, while we describe the specifics of air quality measurement with MOX sensors, the sensor hardware and test platform in Chapter 3. The gathered datasets for our experiments are covered in Chapter 4. In Chapter 5, we present the architecture of used Machine Learning Models and data processing tools to address the above challenges. We evaluate our findings on field data in Chapter 6 and present detailed results. Finally, Chapter 7 concludes our work and gives an outlook into future research directions.

Chapter 2

Related Work

Below, we provide an overview of state-of-the-art miniaturized environmental sensors for mobile and Internet of Things (IoT) applications and discuss challenges preventing their direct usage in Energy Harvesting (EH) - IoT devices. We then present recent work which approaches some of these challenges by using Machine Learning (ML) Models.

2.1 Environmental sensors

Beside human-centric applications such as activity recognition [KLH+17, LXM+19], step counting [RBL+17, KS16], etc. there is a huge interest in deploying large-scale sensor networks to detect states of the environment that may harm human health or cause economic damage. Examples include early warning detection of, *e.g.*, tsunamis [WBF+12], nuclear plant catastrophes [BFB+16] and air pollution [HSW+15]. These applications require the use of environmental sensors, such as those measuring gases and particles. Low-cost sensors measuring gaseous pollutants (*e.g.*, volatile organic compounds (tVOC), carbon monoxide (CO), ozone (O3), oxides of nitrogen (NOx)) appeared on the market less than a decade ago and have a smaller packaging with less power consumption compared to high-end measurement systems. The most popular sensing principles are based on *electrochemical* (EC) or *metal oxide* (MOX) layer reactions. MOX sensors respond to gases with changes in their electrical resistance and are capable of measuring all main gaseous pollutants [FCAB10]. MOX is the most commercially successful type of gas sensor with applications ranging from environmental monitoring [KHK+08, RN09], energy [ALG+15], food [LCM+15], automotive [SBM+09], and safety and security [KPR+06] to biomedicine [RAP15, WB11]. The main limitations of MOX technology are a lack of selectivity, high power consumption, and temporal drift [BM18, PPM+10]. EC sensors contain two electrodes. Gases are either oxidized or reduced at one of the electrodes, causing a potential difference between the two electrodes and thus a current flow [MPS+13, WYZ+10]. The drawback of EC sensors is their long response times in the range of several minutes. Despite the availability of a wide range of portable, low-power and low-cost environmental sensors, a number of challenges prevent their direct usage in EH-IoT devices:

(1) *Long response times.* Chemical reactions take time. Therefore, environmental sensors exhibit *long response times*, often in the range of minutes. High temperatures speed up chemical reactions and allow measuring gases with a higher boiling point, yet at

the price of a considerably higher power consumption.

(2) *Long cold start times.* Environmental sensors age even when they are not used: Gas and humidity molecules accumulate on the sensitive surface, dust particles contaminate the filter of an optical particle sensor, and others. Internal cleaning and baseline adaptation are necessary to compensate for these effects and ensure correct measurements.

(3) *Duty Cycle and on-demand sensor operation cause large measurement errors, significant sensor baseline and sensitivity drifts [BM18].* Most environmental sensors are therefore designed for continuous operation.

2.2 Low-power measurements improved with ML Models

Although sensor manufacturers increasingly add signal conditioning and compensation on chip, these signal manipulations are based on the physical model of the sensor response. Further, these methods can usually not account for uncompleted chemical reactions or sensitivity changes of sensors, which arise with sensors being either duty cycled or operated on-demand. Machine Learning on the other hand promises further optimizations and could allow for low-power applications with accurate measurements.

In [JLZ+18], the authors develop a low-power ammonia monitoring system based on metal oxide sensors and propose to use a LSTM neural network to predict the final chemical equilibrium from a short transient recorded during a $0.2s$ time frame, containing 8 samples. Using LSTM neural networks, an average prediction error rate of 0.12% and a mean absolute error of $9.38ppm$ could be achieved. By shortening the heating time of the sensors, the energy consumption could be brought down by 99.6%. They further show that the model works on data from new sensors, with the average error rate just slightly increasing to 0.20%. Although the authors show feasibility of the approach, they do not mention changes of sensor sensitivity due to different operation modes, necessary to generalize the approach to other gases and MOX sensors.

Recent research in [MK20], duty cycles low cost gas sensors with a 15% duty cycle and analyzes the measurements (transient) during the heating process, instead of waiting for the sensor to fully heat up and reach equilibrium. The sensors are turned on for $20s$ and then sleep for $120s$. Again, a LSTM neural network is used to interpret the transients and then to predict a gas level. The authors state that their approach provides high accuracy for predicting gas levels, while saving up to 85% of energy compared to a sensor which is operated continuously. The approach was further evaluated on two different sensors, with slightly higher errors. The impacts on energy consumption for different on-times are also presented, showing that the gas sensor's impact on energy consumption is the most significant one for high on-times compared to energy needed for a microcontroller controlling the sensor and data transmission, but reducing for smaller on-times.

Considering related work, it is therefore motivating to further investigate into low-power operation of gas sensors especially.

Chapter 3

Hardware

3.1 SGP30 Gas Sensor

The sensor used in our experiments is the SGP30 gas sensor from Sensirion [Sen20c]. It is a digital multi-pixel metal-oxide gas sensor featuring a digital I^2C interface, analog and digital electronics, a temperature controlled micro hotplate and four MOX sensing elements in one single chip. Each of the four MOX sensing elements, also named pixels, are tuned to be sensitive to specific gases, but also remain cross-sensitive to other gases, and can be read out individually. The public interface provides access to the read out of two sensing elements, pixel 1 (P1) tuned for Ethanol and pixel 2 (P2) tuned for Hydrogen (H2), specified in the datasheet. Further details can be found in [Sen20c] and [RHB18]. The Metal-Oxide Gas Sensing principle used by the SGP30 is described in Section 3.2.

The SGP30 sensor offers two preprocessed indoor air quality signals and two sensor raw signals. The two preprocessed indoor air quality signals are tVOC in *ppb* (parts per billion) and CO2-eq in *ppm* (parts per million), which are based on the two sensor raw signal measurements, named Ethanol and H2, in *ticks*. While the sensor technically does not directly measure Ethanol (EtOH) and Hydrogen (H2), the raw signals of pixel 1 and pixel 2 still were named after them, because they were tuned towards these gases. We therefore respect the manufacturer's naming and use Ethanol and H2 to refer to the raw signals. During production, as described in [RHB18], the sensors are calibrated to Ethanol and H2 with individual calibration parameters, so that a conversion from the sensor raw signals to calibrated output signals tVOC and CO2-eq is possible. Additionally, an on-board baseline compensation and humidity compensation (external humidity sensor needed) allow for more accurate measurements. For the raw signals, higher *tick* values correspond to purer air and smaller *tick* values describe worse air quality. For tVOC and CO2-eq, it is the other way around. Small measurement values represent good air conditions, while high measurement values indicate bad air quality levels. It is also worth mentioning that the sensor does not have a *zero / maximum* reading in practice. The highest raw values representing the cleanest air would only be reached in vacuum.

In practice, the SGP30 sensor has to be operated up to a few hours before it can provide reliable measurements, because the sensor has to reach an equilibrium. This is why the manufacturer [Sen20c] recommends to operate the sensor in *Continuous Mode*, which is explained in 3.4, and why the sensor should not be powered off.

3.1.1 Target Gases

tVOC stands for Total Volatile Organic Compounds and describes the total concentration of Volatile Organic Compounds in the air. One pixel (P1) of the SGP30 is calibrated on EtOH, on which the sensor has a similar sensitivity as to the typical gas mixture of tVOC, by computing calibration parameters obtained from the correlation of EtOH and the Ethanol raw signal, which are saved on chip [RHB18]. This allows the sensor to compute an approximation of tVOC in the field based on the calibration parameters and the Ethanol raw signal. Additionally, the sensor applies baseline compensation algorithms for more accurate measurements. This topic is further elaborated in Section 3.7.
The German Federal Environmental Agency related different tVOC concentrations to five indoor air quality levels (IAQ) on a logarithmic scale [20220], which can be seen in Figure 3.1. These values do also correspond to the values provided in Sensirion's Software ControlCenter [Sen20b].

TVOC concentrations and IAQ Levels				
IAQ Level	**TVOC [ppb]**	**Hygienic Rating**	**Recommendation**	**Exposure Limit**
Unhealthy	2200 - 5500	Situation not acceptable	Use only if unavoidable / Intense ventilation necessary	hours
Poor	660 - 2200	Major objections	Intensified ventilation / airing necessary Search for sources	< 1 month
Moderate	220 - 660	Some objections	Intensified ventilation / airing recommended Search for sources	< 12 months
Good	65 - 220	No relevant objections	Ventilation / airing recommended	no limit
Excellent	0 - 65	No objections	Target value	no limit

Figure 3.1: Different tVOC concentrations related to Indoor Air Quality levels and the recommended measures to be taken, presented in [20220].

CO2-eq stands for CO2 equivalent. While CO2 can not be directly measured with the SGP30 sensor, it is expected that indoors there is a correlation between H2 and CO2 concentrations, because both are significantly present in human breath and humans are the main source for CO2 and H2 indoors [RHB18]. Therefore, one pixel (P2) in the SGP30 sensor is designed to have an increased sensitivity to H2, providing the H2 raw signal. As before, this signal can be converted to an equivalent CO2 signal (CO2-eq), by using the sensor's calibration parameters for this pixel saved on chip, which were obtained during factory calibration from the correlation of the H2 raw signal and CO2 values measured with a CO2 sensor, described in [RHB18]. While the pixel providing the H2 raw signal is also influenced by tVOC and variations in H2 and CO2 concentrations in breath, this method still provides a low-cost and good alternative for measuring CO2 levels [RHB18]. In Figure 3.2, one can see the meaning of different CO2 concentrations.

CO_2 concentrations and Sensirion Levels		
Sensirion Levels	**CO_2 [ppm]**	**Details**
Poor	35000	Max. 15 minutes exposure
Poor	5000	Max. CO_2 concentration in workplace for prolonged periods
Moderate	1000	Max. indoor CO_2 concentration acceptable
Good	< 800	Acceptable CO_2 concentration
Good	350 - 450	Typical outdoor concentration

Figure 3.2: CO2 concentrations and details [co2] related to levels from Sensirion's Software ControlCenter [Sen20b].

3.2 Metal-Oxide Gas Sensing Principle (MOX)

The SGP30 gas sensor is based on the Metal-Oxide Gas Sensing principle (MOX), explained in [Sen20a]. Such sensors are also known as Metal Oxide Semiconductor (MOS) gas sensors. A MOX-layer, consisting of metal-oxide particles, is placed between two electrodes on a hotplate as seen in Figure 3.3. In order to obtain accurate measurements, the hotplate has to be heated to high temperatures and the sensor has to reach an equilibrium point, which usually takes some time. The heating of the hotplate to high temperatures results in negatively charged oxygen species at the MOX-layer surface, reacting with the target gas in the ambient air. As a result of the reaction, electrons are released into the MOX-layer, changing its electrical resistance, which is measured between the two electrodes. The change in resistance is directly correlated with the target gas and internally mapped to gas concentrations.

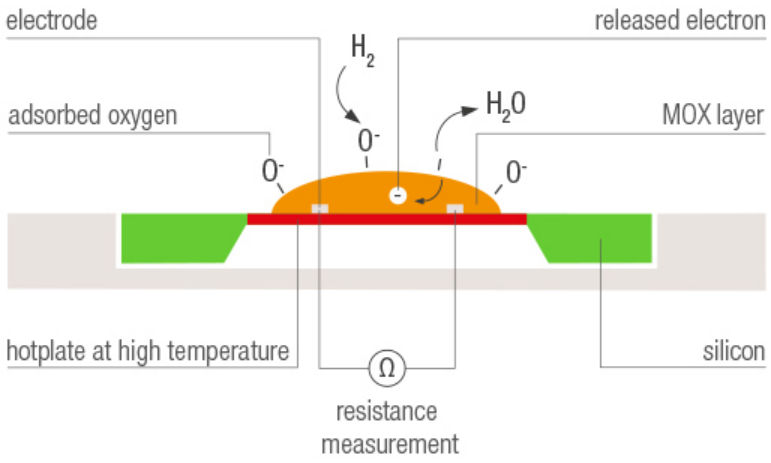


Figure 3.3: Visualization of the Metal-Oxide Gas Sensing Principle (MOX). Image source: [Sen20f]

By changing the composition of the MOX-layer, one can tune the sensor sensitivity to specific target gases, so that the change in resistance reflects the target gas concentration. This technology allows reaching high sensitivity on various gases and is suited for low-cost small package mass production [Sen20a].
Further details and explanations are well described in [PJDP17] and can be read for a more in-depth understanding of this sensing principle.

3.3 Sensor Platform

3.3.1 Build

For easy communication with the sensor and fast prototyping, we have used the SVM30 board from Sensirion [Sen19]. This board includes the SGP30 gas sensor and additionally, a SHTC1 humidity and temperature sensor and power converters, allowing the sensors to be powered from a $5V$ power supply, which is usually available with Microcontrollers or Microprocessors. By using this board, we have on the one hand the advantage that we do not have to design suitable power conversions for the SGP30 sensor, which should be operated at $1.8V$, as they are directly incorporated into the board and on the other hand, we get an additional sensor for measuring humidity and temperature, which can give additional insight into the air quality.

In order to work with the sensors, we have designed a custom build. On a wood plate, we mount three SVM30 boards, where each board can be used for different operation modes, which are described in detail in Section 3.4. The main application is that one sensor board is running in *Continuous Mode* and providing ground truth values, one sensor board is running in *Duty Cycle Mode* and the other sensor board is running in *On Demand Mode*. These three sensors are placed inside a plastic frame, which can be closed. This allows to introduce various events changing the air quality and to keep the air concentration more or less stable inside the plastic box, when necessary. Additionally, we mount a small fan in the vicinity of the sensors, which can be turned on when needed.

For each of the three sensor boards, we mount one Raspberry Pi 3 Model B [FOU20] to the wood plate. We use one Raspberry Pi for each sensor board separately, because we want to have independent setups and avoid using I^2C multiplexers in order to keep the build complexity as low as possible. This procedure did not impose any complications until the end and every experiment could be performed as expected. Between the three sensor boards and the three Raspberry Pis, we mount a breadboard so that we can wire up everything. For the functionality to turn the sensors on and off using GPIO pins of the Raspberry Pis, which is needed for different operation modes, we integrate three BC547B NPN transistors from ON Semiconductor [ON 12] on the low side. In Figure 3.4 one can see the completed build of the platform used for our experiments.

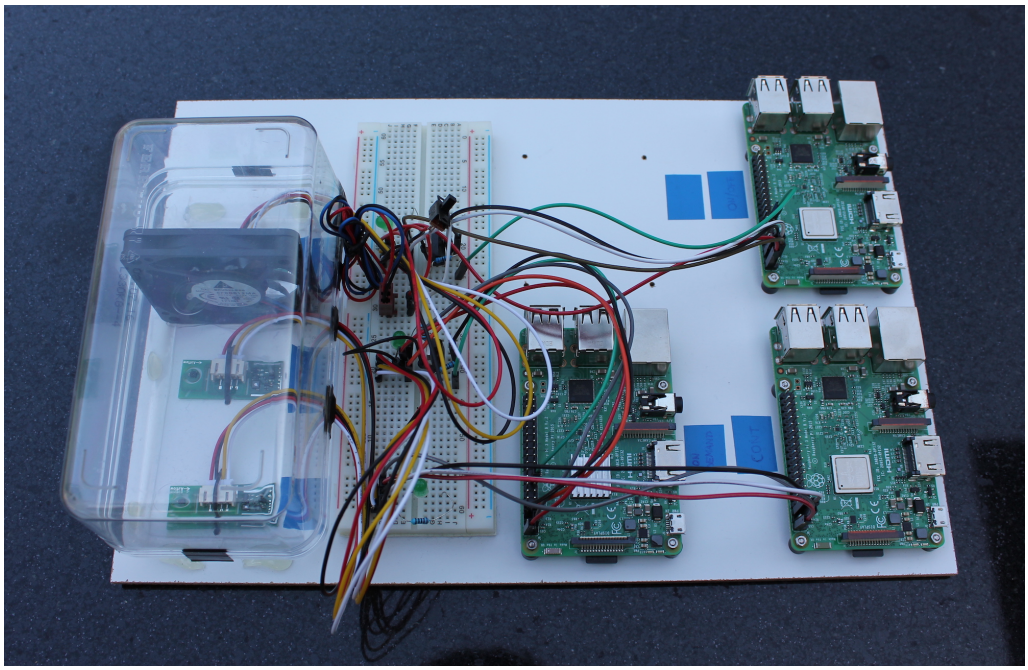


Figure 3.4: The Sensor Platform used for performing the experiments.

The SVM30 sensor board has a connector with four outlets: SCL, GND, VDD, SDA. SCL and SDA are directly connected to the corresponding pins on the Raspberry Pi, which are pin 5, also named GPIO 3 (SCL) and pin 3, also named GPIO 2 (SDA), respectively. The VDD outlet of the SVM30 sensor board is connected to the 5V power line of the Raspberry Pi. The GND outlet of the sensor board is connected to the collector of the transistor. The emitter of the transistor is connected to GND of the Raspberry Pi. In order to control the transistor, we connect its base in series with a $2.2k$ to pin 15, named GPIO 22, of the Raspberry Pi. A schematic of the connections can be seen in Figure 3.7.

In order to have reproducible results and to have the same ground truth for all measurements, we assign each sensor a specific role. We choose one sensor board, named *Device 1*, to be the sensor board which is always operated in *Continuous Mode*. As this is the recommended operation by the manufacturer and we can expect the best possible measurements from this sensor, we treat its measurements as the ground truth measurements against which we compare other measurements and outcomes of our predictive models. One sensor board, named *Device 2*, is operated in *On Demand Mode*. The last sensor board, named *Device 3*, is operated in *Duty Cycle Mode*. The different operation modes are explained in Section 3.4 in detail.

3.3.2 Taking Measurements

In this section we describe the procedure necessary to perform a measurement with the SVM30 sensor board and the Raspberry Pi. First of all, we have to put the pin, which is controlling the transistor, to HIGH, so that enough current can flow through the transistor allowing the sensor board to operate properly. We now have to wait for the SGP30 sensor to power up, which takes up to $0.6ms$. Then, we initialize the

I^2C communication, with the sensors I^2C address being $0x58$, using the Python library *Adafruit_CircuitPython_BusDevice* from adafruit [ada20a]. After turning on the sensor and setting up the communication, we can start with the measurements using the available commands described in the sensor's datasheet [Sen20c]. There is a library available from adafruit [ada20b], which implements the driver for the SGP30, but we have written a custom software in order to perform measurements and to communicate with the sensor, because the commands to read the raw values were not available in the previously mentioned library. Nevertheless, we could use the code from the library as a basis for our code and in the end, we also contributed to the SGP30 driver library from adafruit and implemented the reading of raw values, so that other developers can use this functionality.

The first command which we send is the `sgp30_iaq_init` command (max. duration = 10ms), which starts the air quality measurement. After this command, we can send the `sgp30_measure_iaq` command (max. duration = 12ms) in order to get the air quality signals tVOC and CO2-eq or the `sgp30_measure_raw` command (max. duration = 25ms) in order to get the sensor raw signals Ethanol and H2. The commands to retrieve measurements can be repeated as long as one wants to measure and can be sent in arbitrary time steps. Nevertheless, to ensure proper operation, it is recommended by the manufacturer to send the `sgp30_measure_iaq` command in regular intervals of 1 second. This is done for the sensor running in *Continuous Mode*, where we first send the `sgp30_iaq_init` command and then, in regular intervals send the `sgp30_measure_iaq` command, directly followed by the `sgp30_measure_raw` command. This allows us to read Ethanol and H2, as well as tVOC and CO2-eq, regularly.

When operating the sensors in *Duty Cycle Mode* or *On Demand Mode*, we are interested in the behaviour of the measurements in the first few seconds when the sensor is on, the transient response, and not in the long term behaviour. Further, tVOC and CO2-eq are not provided during the first 15 seconds of measurement and as we are interested in measurement times smaller than that, we solely focus on the sensor raw signals in these operation modes. In *Duty Cycle Mode* and *On Demand Mode*, we want to get as many measurements as possible, and thus the most information, during the time the sensor is turned on, which is why we do not send the measurement command in regular intervals of 1 second, but we send the command as often as possible. With the maximum duration of $25ms$ for one `sgp30_measure_raw` command, we can theoretically read the sensor raw signals with a maximum rate of $40Hz$. Due to delays coming from the sensor turn on, the communication and the initialization of the sensor, we manage to reach a raw signal reading rate of $36Hz$ in practice, which is sufficient for our experiments. In the future, the code and build can be optimized if necessary. Once we are done with the experiment and we want to turn the sensor off, we just have to put a LOW signal with the Raspberry Pi to the transistor's base. The details and functionality of the sensor are present in the SVM30 [Sen19] and SGP30 [Sen20c] datasheets.

3.4 Operation Modes

Each sensor board is operated in a specific operation mode, that we manually configure. The operation modes are *Continuous Mode*, *Duty Cycle Mode* and *On Demand Mode* which we will describe in detail in the following.

3.4.1 *Continuous Mode*

When a sensor board is operated in *Continuous Mode*, it means that the sensor is continuously powered with enough energy and the sensor is constantly in measurement mode, with the hotplate at high temperatures. This allows the sensor to reach an equilibrium state. Measurements are read in regular intervals, as recommended in the datasheet, which ensures that the measurements are as accurate as possible. Therefore, the values obtained in this operation mode are seen as ground truth values, against which we compare other measurements and outcomes of our predictive models, as already described. In this operation mode, we read Ethanol, H2, tVOC, CO2-eq from the SGP30 and Temperature and relative Humidity from the SHTC1. Generally, this operation mode consumes the most energy, as the sensor is constantly on and the hotplate is kept constantly hot.

3.4.2 *Duty Cycle Mode*

Due to the fact that gas sensors need respectively high power to work, they are not suitable for mobile devices or battery-less implementations in the IoT-domain, as that amount of energy is either not available or too precious to be used for gas sensors. It is therefore very important to find methods to reduce the energy consumption of gas sensors in order to use them in low power applications.

A common way to reduce power consumption is duty cycling. In this operation mode, named *Duty Cycle Mode*, one defines a time frame T_{on}, for which the sensor is turned on and another time frame T_{off}, for which the sensor is turned off. The time it takes a system to run an on and off phase is called *period*, giving $T_{period} = T_{on} + T_{off}$. Duty cycling usually is expressed in % as a ratio between the on-time and the period [dut] with

$$D = \frac{T_{on}}{T_{period}} \times 100\% \tag{3.1}$$

Because of the fact that the sensor now is not continuously powered as the manufacturer advises, we will observe unexpected measurement results. The question arises, whether it is possible to create models based on the measurements obtained in *Duty Cycle Mode* to compute reliable and accurate measurements as one would get from a sensor operated in *Continuous Mode*. Other research has shown in [MK20] and [JLZ+18], that there is valuable information in the first measurements of a gas sensor, also known as transient response, which can be used by models to predict an accurate measurement. In this operation mode, we are therefore interested in the transient response. The on-time of the sensor and thus the length of the transient response impacts the energy savings achieved with duty cycling. Early values of the transient are highly affected by relative humidity change rather than tVOC, which is why the length of the transient is important for model creation. Further, the challenge we are dealing with in this operation mode is the changing sensitivity of the sensor due to it not being constantly on and particles in the air and incomplete chemical reactions changing the properties of the sensor's surface.

In our work, we investigate the effects of very small duty cycling, below 2%, with an on-time of up to 5 seconds. Sensirion also investigated duty cycling with the SGPC3 [Sen20e], a sensor which is always duty cycled, with an on-time of $40ms$ and periods of $2s$ or $30s$. Note that the tVOC measurements from SGPC3, which is always duty cycled, and SGP30 in *Continuous Mode* deviate when operated in parallel in the field due to a

changed sensor sensitivity. Due to our small duty cycle for the SGP30, the sensor will be mostly off, by which we can save a lot of energy compared to it being operated in *Continuous Mode*. During the on-time of the sensor, we take as many measurements for Ethanol and H2 as possible. In this mode, the measurement of tVOC and CO2-eq does not work, because these values are only available after 15 seconds from the sensor. Further Temperature and relative Humidity are only measured at the end of the on-time, so that we can get the maximum amount of measurements within the on-time, in order to have maximum information available.
When the sensor is powered on, the hotplate inside starts heating up and various chemical reactions on the sensor's surface begin. One of the first reactions is water evaporation on the sensor's surface. During this process, we already collect measurements from the sensor for the specified on-time, giving the transient response. These measurements are influenced by many effects, coming for example from the sensor's surface current state, the heating of the hotplate and the different chemical reactions. As already mentioned, we can sample measurements with approximately $36Hz$ and thus manage to retrieve 180 values during an on-time of 5 seconds. The length of the transient response is denoted as W, where for example $W40$ denotes the first 40 values of the transient response, representing around 1 second of the on-time. The transient responses of a sensor operated in *Duty Cycle Mode* from consequent measurement phases during a period, where the air quality did not change, can be seen in Figure 3.5.

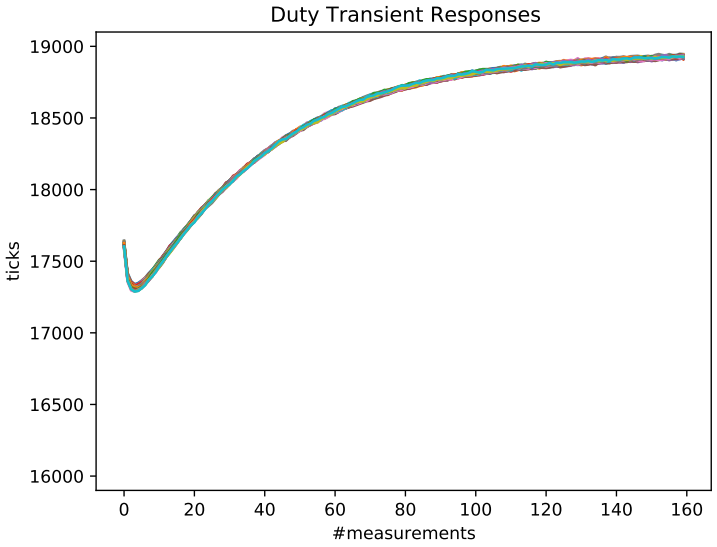


Figure 3.5: *Duty Cycle Mode* Transient Responses from consequent measurement phases during constant air quality.

We can see, that within a transient response, the measurement values increase over time. In addition, the transient response in *Duty Cycle Mode* exhibits a high degree of repeatability, as it is nearly the same for every duty cycle measurement, given the background air composition does not change.

As already stated, we investigate, whether it is possible to predict an accurate mea-

surement, as would have been obtained by a sensor in *Continuous Mode*, by using parts of or the whole transient response of a sensor operated in *Duty Cycle Mode*. We will also investigate, how many measurements from the transient response are necessary for a good performance.

3.4.3 *On Demand Mode*

On Demand Mode is in principle exactly the same as *Duty Cycle Mode* explained before, except that the sensor is turned on aperiodically. Usually, this type of operation is applied when energy is not constantly available and one performs a measurement as soon as energy is available, so the sensor is operated on demand. This behaviour allows for equal on-times T_{on}, but the sensor off-times T_{off} are non-equal, unpredictable and dependent on the energy availability.

The transient responses from consequent measurement phases in *On Demand Mode* during the same time span as seen in Figure 3.5, where the air quality did not change, can be seen in Figure 3.6.

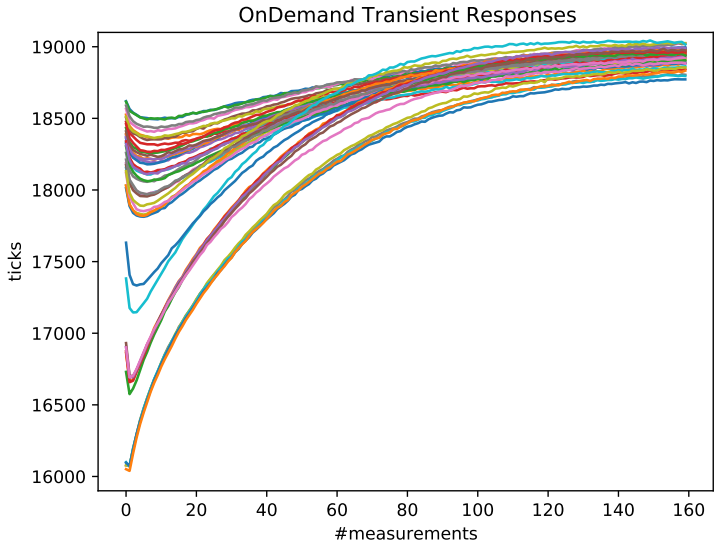


Figure 3.6: *On Demand Mode* Transient Responses from consequent measurement phases during constant air quality.

We can see again, that the measurement values of a transient response increase over time. But in contrast to a sensor operated in *Duty Cycle Mode*, the transient responses from a sensor operated in *On Demand Mode* are not the same for consequent periods, given unchanged air quality. The transient responses are highly dependent on and influenced by the past off-times of the sensor, which could be due to the fact, that during different off-times, the sensor's surface is influenced differently by components in the ambient air. This imposes additional challenges, which need to be considered in model creation. We will investigate, whether it is possible to correct for the influence of varying off-times on the transient responses and whether it is possible to understand how to still get accurate

measurements, even though the sensor is operated on demand.

3.5 Energy Consumption

One important characteristic of sensors is their energy consumption. As already mentioned, sensors based on the Metal-Oxide Gas Sensing principle have a high energy consumption compared to other sensors, like for example temperature sensors or accelerometers. We propose an approach to reduce the energy consumption by operation in *Duty Cycle Mode* and applying ML models.

With the data provided in the SGP30 and SVM30 datasheets [Sen20c, Sen19], we calculate the energy consumption for different on-times of the devices, which can be seen in Table 3.1. Nevertheless, we also measured the energy consumption for the SVM30 sensor board in practice by using the RocketLogger [SGL+16]. The RocketLogger can be connected to our sensor, as seen in Figure 3.7 and performs voltage and current measurements, which allows to also calculate the energy consumption. This device is easy to use and offers a browser based graphical interface, where different measurement settings can be applied.

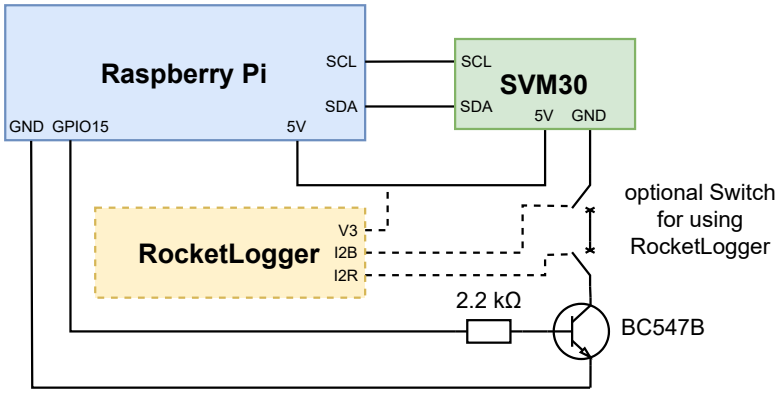


Figure 3.7: Schematic showing the connections of one unit on the sensor platform, including one Raspberry Pi, one SVM30 sensor board, one transistor and one resistor. For performing measurements, a RocketLogger can be connected optionally.

We performed measurements of the SVM30 sensor board, where we periodically turned it on for 2 seconds and then turned it off for 10 seconds, so that we could investigate the sensor's *on* and *off* behaviour. One sample of these measurements can be seen in Figure 3.8.

When the sensor board is turned on, the current is noticeably higher for a short period of time. This is an expected behaviour, as the datasheet states that the current draw is 20% higher in the first $5ms$ in the measurement mode. The mean energy consumption based on these measurements for the SVM30 is $0.469J$, the mean voltage $4.92V$ and the mean current draw $0.047A$. These values are slightly lower than stated in the datasheet, but are in the typical range and influenced by various factors, like power supply voltage and sensor variation. Using these measurements, we could validate our calculations based on the values from the datasheets. We then also calculated energy consumptions for other

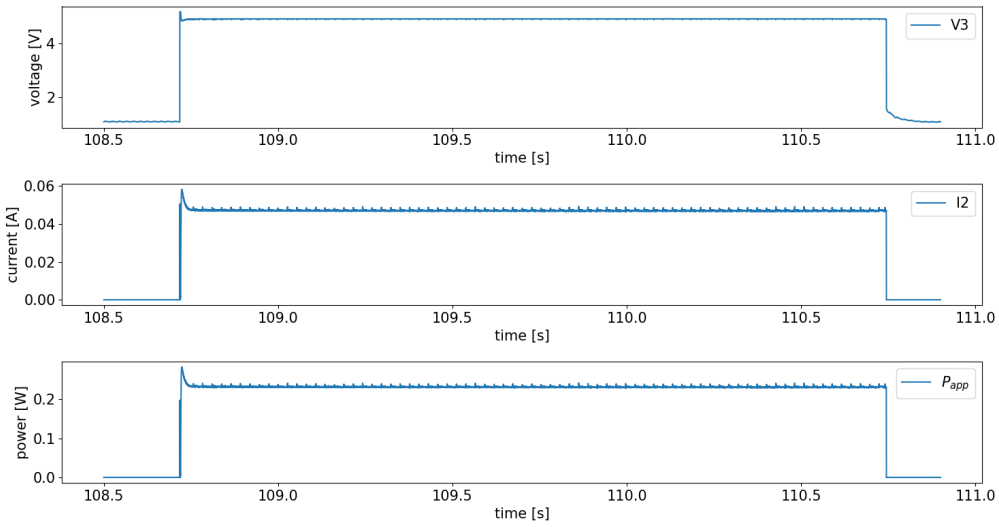


Figure 3.8: Voltage, Current and Power measurement of SVM30, with 2 seconds on-time, using RocketLogger.

on-times. The calculated energy consumptions for SGP30 and SVM30 with different on-times can be seen in Table 3.1. Note that we also included the SGP40 sensor [Sen20d], the successor of the SGP30, which was released towards the end of our work and promises to be an interesting sensor in the context of battery-less sensing based on the same principle as the SGP30, but with a lower energy consumption.

Device	*Voltage [V]*	*Current [A]*	*1s ON*	*2s ON*	*5s ON*	*10s ON*
SVM30	5.0	0.049	0.245 J	0.490 J	1.225 J	2.450 J
SGP30	1.8	0.048	0.086 J	0.173 J	0.432 J	0.864 J
SGP40	1.8	0.0035	0.006 J	0.013 J	0.032 J	0.063 J

Table 3.1: Calculated energy consumptions for different devices and different on-times.

It is noticeable that the SGP30 has a smaller energy consumption than the SVM30. This can be explained with the fact, that the SVM30 sensor board includes additional hardware for voltage regulation beside the SGP30 sensor. We have included the values for SGP30, because for a practical application, one would prefer the SGP30 and design the power supply with regard to the application and not use the SVM30 sensor board, as it is intended for prototypes.
While the energy consumptions obtained with the SGP30 are already low, the SGP40 sensor promises far lower energy consumptions. In the future, it would be therefore very interesting to further analyze the differences in the sensor characteristics between SGP30 and SGP40 and to evaluate the possibilities of the SGP40 for low power applications.

3.6 Sensor Differences

When working with sensors of the same type, like we do in this thesis with three SGP30 sensors, in theory one would expect that when running simultaneously in the same operation mode, all sensors would provide the same measurements. In practice, when working with sensors already operated in different environments or operation modes, this is not the case, as there will always be differences across the sensors. The surface in gas sensors, which is reacting with the surrounding air, changes its properties depending on how and how often or long it was operated, where it was stored and many more. These influences have an impact on the sensor measurements and are the reason why it is not possible to have several equal sensors which all provide exactly the same measurement value. Because we are working with three equal sensors, this behaviour is interesting for us, which is why we wanted to look into it further. We wanted to know, how two of the sensors, *Device 2* and *Device 3*, behave with respect to the other one, *Device 1*, which is always operated in *Continuous Mode* and whose measurements are seen as the ground truth. For this, we created an experiment, where we operated all three sensor boards in *Continuous Mode* for 60 hours. In Figure 3.9 one can see the sensor raw signals over time for *Device 1* and *Device 2* and in Figure 3.10 for *Device 1* and *Device 3* respectively.

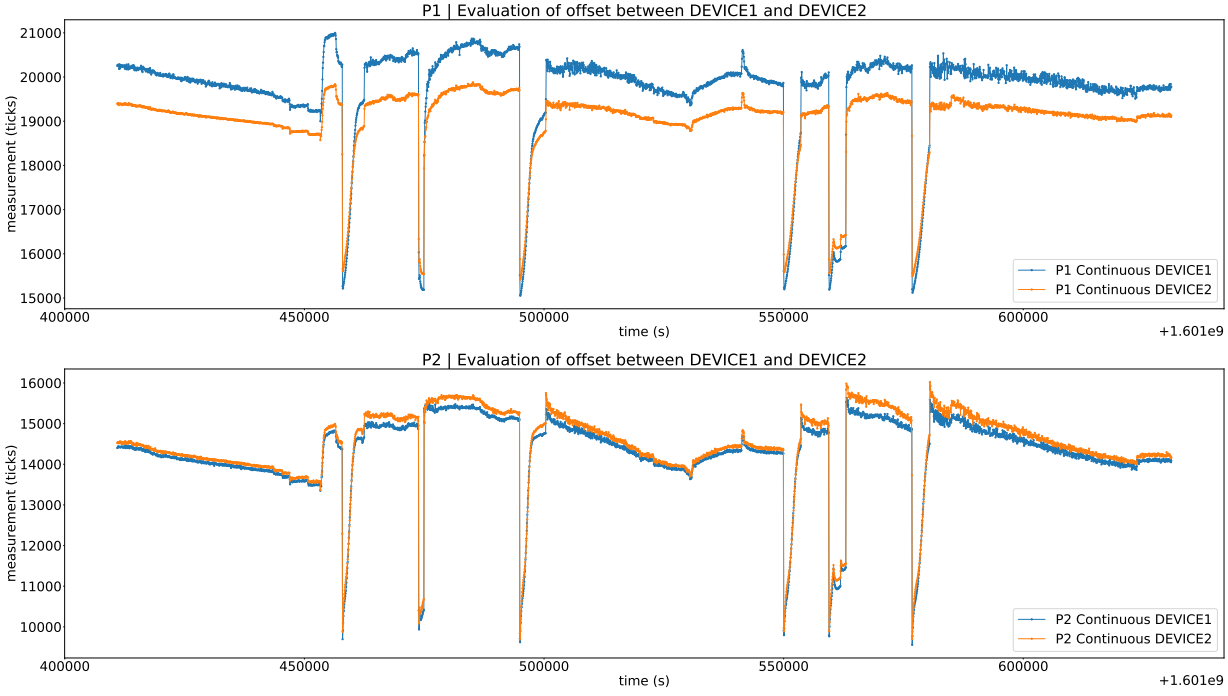


Figure 3.9: *Device 1* and *Device 2* operated in *Continuous Mode* for 60 hours to visualize their offset to each other.

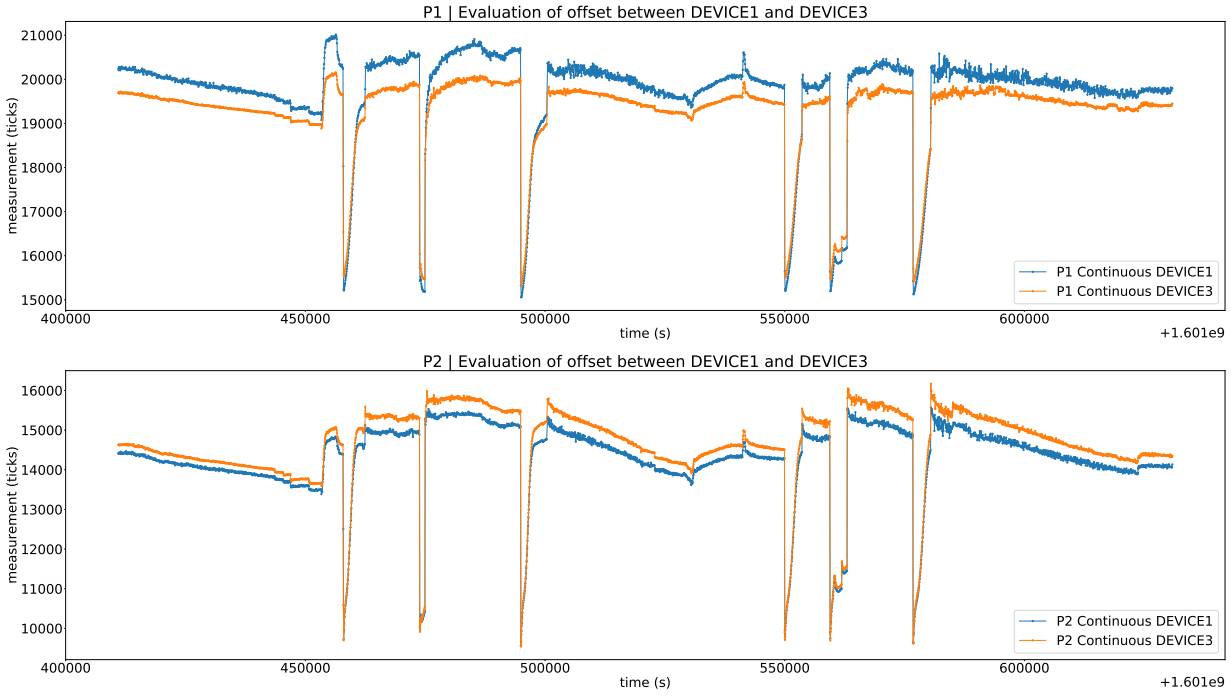


Figure 3.10: *Device 1* and *Device 3* operated in *Continuous Mode* for 60 hours to visualize their offset to each other.

We can observe that while measuring under the same air quality conditions, the three sensors do not provide the same measurement values. The measurement values are offset to each other with a non constant factor, which also is influenced by the height of the measurement value. We have repeated this experiment after five days and observed the same results, where the measurement values again reflect an non-constant offset. From this experiment, we learn that Machine Learning Models do not only need to predict an accurate measurement from a transient response, but also need to account for these offsets.

3.7 Measurement Units: From *ticks* To *ppb/ppm*

Beside the fact that it is important to have accurate measurements, it is also important that the obtained measurements can be interpreted. In case that the sensors are operated in *Duty Cycle Mode* or *On Demand Mode*, we do not have measurements for tVOC or CO2-eq, but only raw signal measurements with a good sensitivity to EtOH and Hydrogen. While tVOC is measured in parts per billion (*ppb*) and CO2-eq in parts per million (*ppm*) and therefore can be interpreted and mapped to specific air quality levels, the raw signals on the other hand are provided in *ticks*, a unit which is not directly interpretable. We therefore need to apply a conversion from *ticks* to *ppb* or *ppm*, so that we can draw conclusions with respect to air quality levels.
The datasheet of the SGP30 sensor [Sen20c] offers a formula to calculate some gas con-

centration c relative to a reference concentration c_{ref} with

$$c = c_{ref} \cdot exp(\frac{s_{ref} - s_{out}}{512}), \tag{3.2}$$

where s_{out} describes the sensor raw signal in *ticks* at concentration c and s_{ref} describes the sensor raw signal in *ticks* at the reference concentration c_{ref}. c_{ref} and s_{ref} are calibration parameters, which are evaluated in factory during production respectively for both pixels, P1 and P2, by exposing the sensor to a known reference concentration c_{ref} and then reading the sensor raw signal giving s_{ref} [RHB18]. These parameters, which are unique for each individual sensor, are then once saved on chip, and used for direct on-chip calibration to map measurements in *ticks* to *ppb* for tVOC and to *ppm* for CO2-eq respectively [Sen20c]. Additionally, the SGP30 applies internal baseline compensation algorithms to these values to further improve the tVOC and CO2-eq measurements.
As c_{ref} and s_{ref} can not be read out by the user, we can not directly use the given formula. Additionally, the baseline compensation algorithms are not described, which is why we do not know how the concentration values are further influenced. Nevertheless, we still want to be able to convert our sensor raw signal measurement in *ticks* to some tVOC value in *ppb*. We therefore recorded data for each of our three devices running in *Continuous Mode*. We then took, for each device respectively, the tVOC measurements and computed the c_{ref} and s_{ref} values, with which we could then use the formula from above, and calculate concentrations c, so that the difference between tVOC and c is as small as possible. As already stated, we can not account for the influence coming from the baseline compensation algorithm, but this method still allows us to compute concentrations c based on measurements in *ticks* very well. In Figure 3.11, one can see the alignment of tVOC and c for *Device 1*, by using the best possible c_{ref} and s_{ref}. We can see, that both signals align pretty well, which is why this method is accurate enough for our experiments in order to be able to give statements regarding values in *ppb* and air quality levels. The same process can be applied analogously for the raw signal H2 in *ticks* and CO2-eq in *ppm* with pixel 2 (P2).

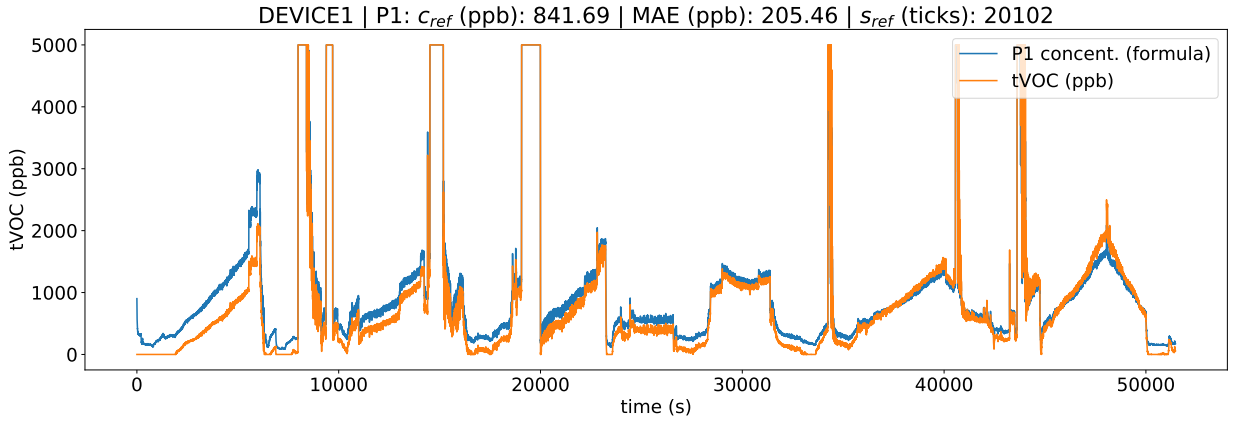


Figure 3.11: Alignment of tVOC and concentration c computed using best computed c_{ref} and s_{ref}, for *Device 1*. High values were capped to 5000*ppb* for better visualization.

Chapter 4

Data Collection

During our work, we have performed several real world experiments with the sensors, collected and recorded measurements and constructed different datasets. These datasets are used to train and evaluate different models on their performance, to gain additional insights into the sensor measurements and how they can be used.

We have focused our work on a few chosen datasets, which contain enough information for our analysis and model creation. These datasets, on which we will also reference in other sections, are named:

- $D_{train+test}$ and D_{test}^{+4d}
- D_{test}^{+18d}
- D_{test}^{+22d}
- D_{test}^{+53d}

The subscript indicates the usage of the dataset and the superscript indicates the days (d) passed after the end of the first dataset $D_{train+test}$, for each dataset respectively. For all these datasets, the setup of the build was similar and the devices were running simultaneously in their respective operation mode:
Device 1 was running in *Continuous Mode* and we have read measurements from it regularly.
Device 2 was running in *On Demand Mode*, with a fixed on-time T_{on} of 5 seconds and a random period T_{period}, which was chosen randomly from $[30, 45, 60, 90, 120, 300, 600, 1200]$ seconds. For D_{test}^{+53d}, we have removed the period of 1200 seconds.
Device 3 was running in *Duty Cycle Mode*, with a fixed on-time T_{on} of 5 seconds and a fixed period T_{period} of 300 seconds (5 minutes).

The number of samples in each dataset and for each sensor can be seen in Table 4.1. For the sensor operated in *Continuous Mode*, one sample means one measurement for each regular interval. For the sensors operated in *Duty Cycle Mode* and *On Demand Mode*, one sample represents one transient response, meaning all the sensor raw signals read within one specific on-time T_{on}.

	Operation Mode	*#Samples*	*Duration*	*Peculiarity*
$D_{train+test}$	*Continuous*	51485	144 hours	for training, with events
$D_{train+test}$	*On Demand*	1700		
$D_{train+test}$	*Duty Cycle*	1728		
D_{test}^{+4d}	*Continuous*	34325	96 hours	for evaluation, with events
D_{test}^{+4d}	*On Demand*	1099		
D_{test}^{+4d}	*Duty Cycle*	1152		
D_{test}^{+18d}	*Continuous*	22768	63 hours	for evaluation, no events
D_{test}^{+18d}	*On Demand*	780		
D_{test}^{+18d}	*Duty Cycle*	764		
D_{test}^{+22d}	*Continuous*	27133	72 hours	for evaluation, no events
D_{test}^{+22d}	*On Demand*	949		
D_{test}^{+22d}	*Duty Cycle*	911		
D_{test}^{+53d}	*Continuous*	326164	96 hours	for evaluation, no events
D_{test}^{+53d}	*On Demand*	1944		
D_{test}^{+53d}	*Duty Cycle*	1152		

Table 4.1: Detailed information about the recorded datasets.

In the following, we would like to further present some important details and plots of the mentioned datasets. In the plots, we show the raw signals Ethanol, on top, and H2, on bottom, over time for all three sensors. For the sensors running in *Duty Cycle Mode* and *On Demand Mode*, we plot for each sample the last value of the transient response, because this measurement represents the best value available, as the sensor was turned on the longest for it. In addition, we plot the temperature and humidity, which was measured by *Device 1* operated in *Continuous Mode*, but was not used in our experiments. In Figure 4.1 we provide a timeline of when the datasets were recorded, their time spans and the breaks between them.

Figure 4.1: Timeline for the datasets, showing their lengths and breaks between them.

4.1 $D_{train+test}$ & D_{test}^{+4d}

$D_{train+test}$ contains the data, which we split into a train and test set in order to train and evaluate the model, and was taken over a time span of 6 days. D_{test}^{+4d} was recorded after a 4 days break over a time span of 4 days, where all sensors were turned off, in order to have data to further evaluate the trained model. In Figure 4.2 and 4.3 one can see plots of $D_{train+test}$ and D_{test}^{+4d}. During the recording of these two datasets, we occasionally introduced artificial events by placing a source evaporating alcohols in vicinity of the sensors for a certain period of time to ensure the models also learn to deal with increased

pollutant concentrations. These events are reflected through a large change of values in the measurements, also visible in the plots. We have to note that such extreme events do not really happen in reality, which is why we also recorded datasets without them.

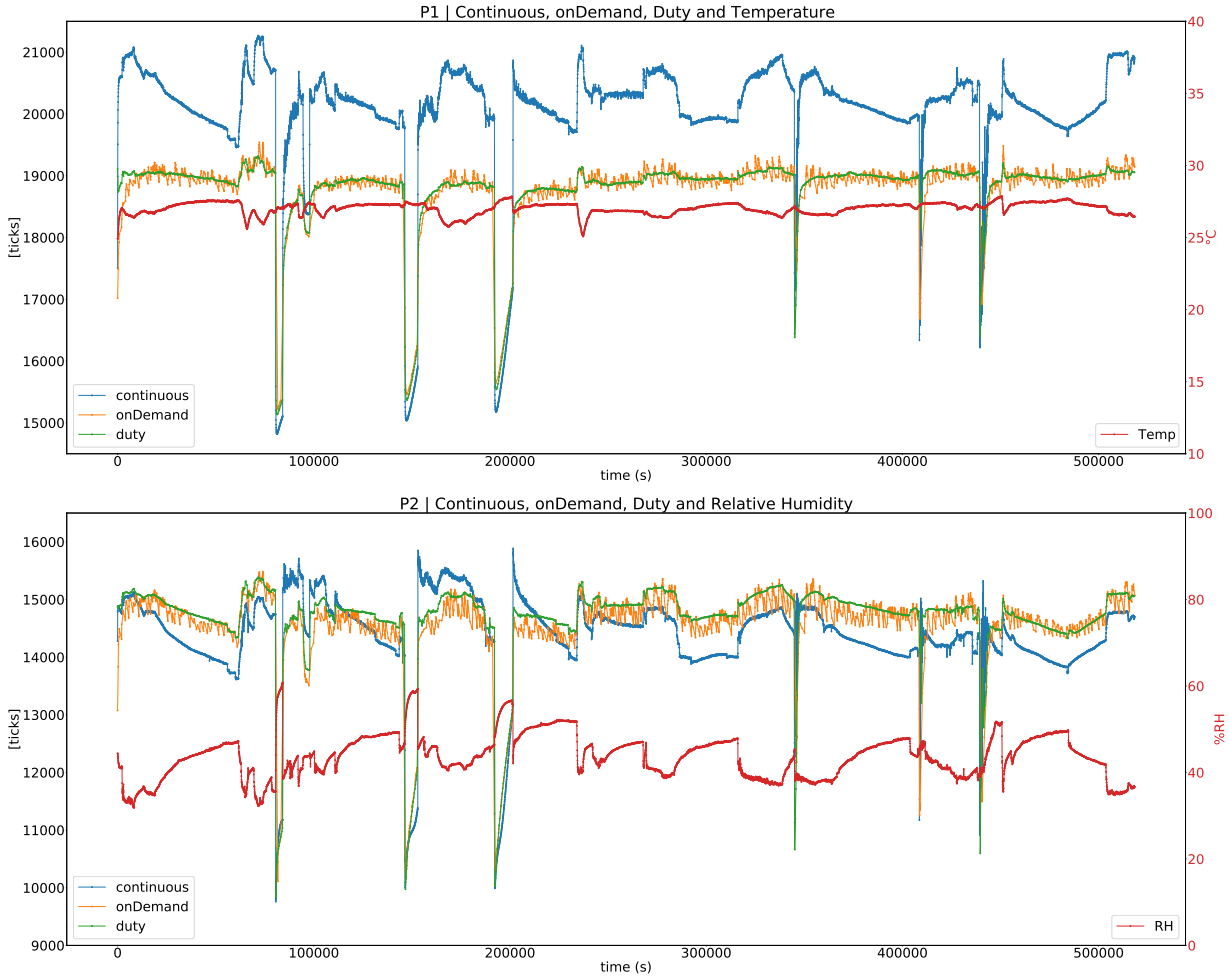


Figure 4.2: Plot showing the sensor raw signals Ethanol and H2 from $D_{train+test}$ for *Device 1*, *Device 2* and *Device 3*.

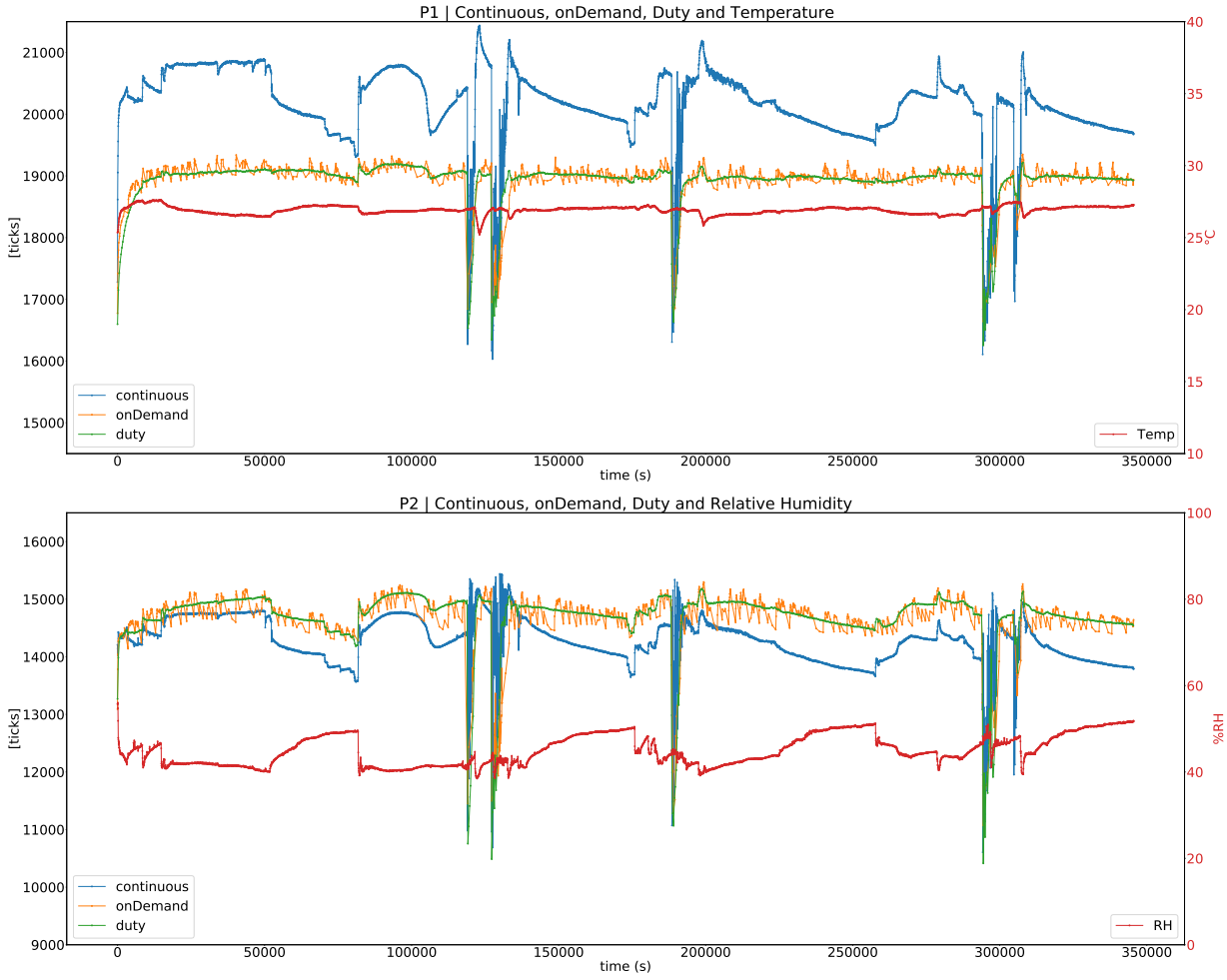


Figure 4.3: Plot showing the sensor raw signals Ethanol and H2 from D_{test}^{+4d} for *Device 1*, *Device 2* and *Device 3*.

4.2 D_{test}^{+18d}

D_{test}^{+18d} was recorded over a span of 63 hours to obtain another dataset, on which to evaluate the trained models and which is taken farther in time than the previous datasets. It was captured 18 days after the end of $D_{train+test}$, which was used to train the models, and can be seen in Figure 4.4. We did not introduce any events during the recording of this dataset.

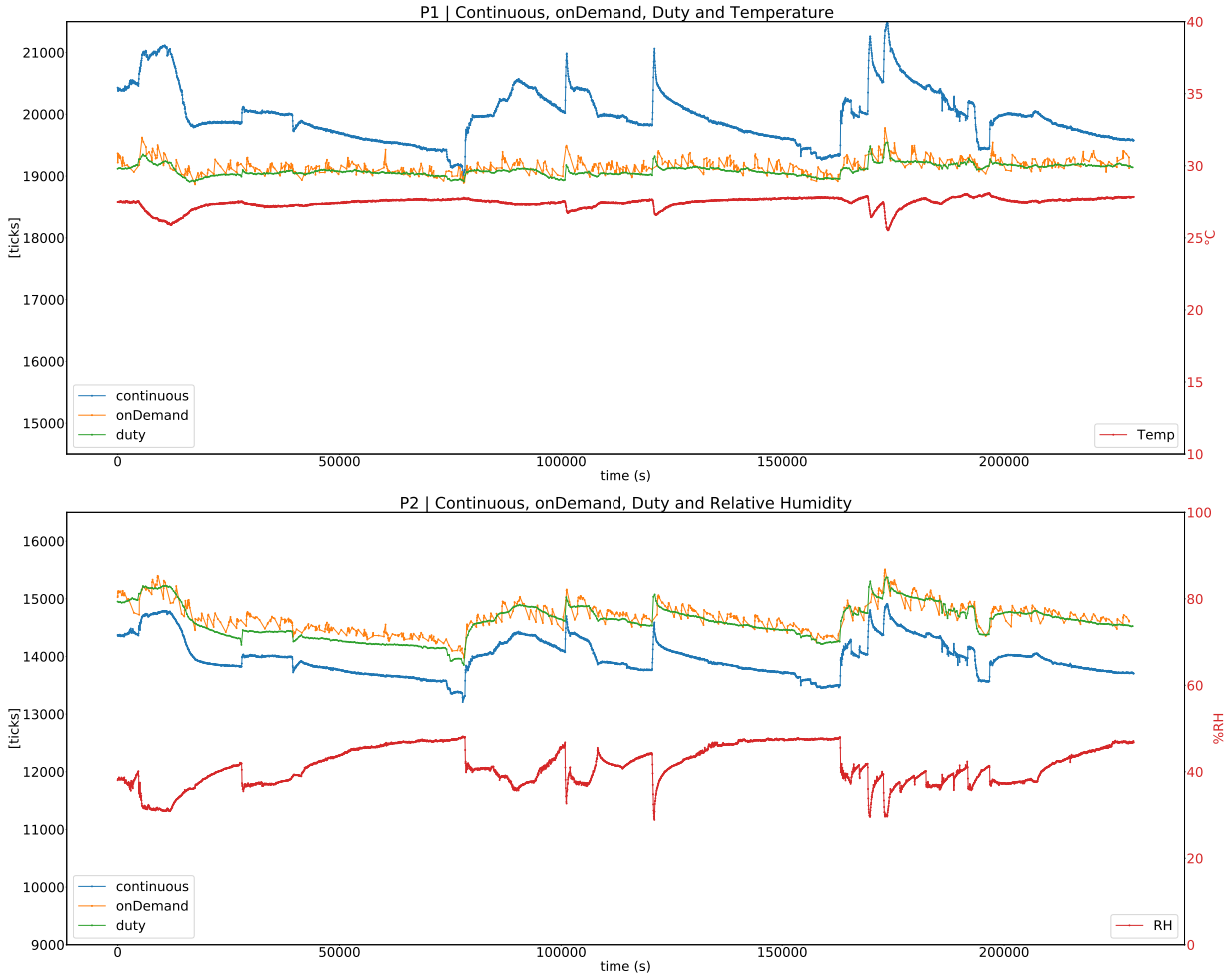


Figure 4.4: Plot showing the sensor raw signals Ethanol and H2 from D_{test}^{+18d} for *Device 1*, *Device 2* and *Device 3*.

4.3 D_{test}^{+22d}

D_{test}^{+22d} was captured over a time span of 3 days, 22 days after the end of $D_{train+test}$ and will also be used to evaluate the trained models. We did not introduce any events during the recording of this dataset. D_{test}^{+22d} can be seen in Figure 4.5.

Figure 4.5: Plot showing the sensor raw signals Ethanol and H2 from D_{test}^{+22d} for *Device 1*, *Device 2* and *Device 3*.

4.4 D_{test}^{+53d}

D_{test}^{+53d} was captured over a time span of 4 days, 53 days after the end of $D_{train+test}$, without introducing any events. For *Device 2*, operated in *On Demand Mode*, we did remove the value 1200 *seconds* from the list representing the possible periods, as already mentioned before. D_{test}^{+53d} can be seen in Figure 4.6.

Figure 4.6: Plot showing the sensor raw signals Ethanol and H2 from D_{test}^{+53d} for *Device 1*, *Device 2* and *Device 3*.

Chapter 5

Machine Learning Models & Data Preprocessing

5.1 Machine Learning Models

In this section, we would like to present the Machine Learning (ML) models we have used in this thesis. We will shortly describe their theoretical background, their special properties and how they work. The information provided in this section is inspired by the articles written in [Den15], [Chr15] and [Mic18].

5.1.1 RNN

The transient response of a sensor operated in *Duty Cycle Mode* or *On Demand Mode* can be seen as a time series of measurements, which are related to each other and can not be seen as independent from each other. The measurements follow a certain pattern, depending on air quality, on-time, sensor surface state and others. It is therefore crucial for a Machine Learning Model to also learn the relationship between the measurement values of a transient response in order to make a good prediction. Traditional Machine Learning Models, like for example Multilayer Perceptrons (MLP), a feedforward artificial neural network, are not specially designed to detect relationships between different values in time series. For this kind of problem, RNNs, Recurrent Neural Networks, are suitable. An RNN has a looping mechanism that allows information to persist and information to flow from one step to the next, which allows the network to memorize important details. An RNN takes a time series as an input and one can read one or more hidden states as the output. The structure of an RNN can be seen in Figure 5.1.

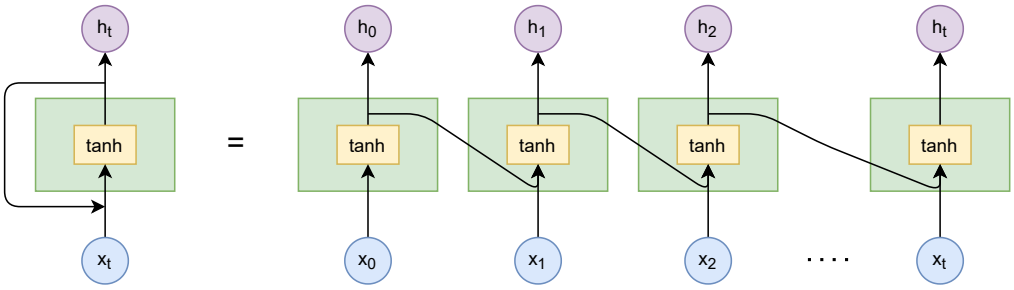


Figure 5.1: RNN structure visualization showing the looping mechanism and its unrolled form. Drawing inspired by [Chr15].

The information flowing from one RNN cell to the other is called hidden state and is a representation of all previous inputs. This kind of network consists of multiple equal simple networks, where each gives some information to the next one. As RNNs can work with multiple inputs, it is best suitable for time series data or sequences. When recent information contains most of the relevant information, which is needed for a good prediction, RNNs work pretty well. But for cases, where crucial information in time series or sequences is encoded in more earlier input values, the RNN struggles to learn these relationships, so called long term dependencies. This problem is called Short Term Memory and is a result of the "Vanishing Gradient Problem", which is also prominent in other neural network architectures.

A neural network usually learns because the weights are adjusted using gradients through a method called backpropagation. The bigger these gradients are, the bigger the adjustment of the weights. The problem is, that these gradients shrink with each layer and for RNNs, the gradients shrink exponentially. This means that early layers, which represent early information, will not learn enough or nothing at all, because of really small gradients. This is the reason why this problem is called the "Vanishing Gradient Problem". In practice, it is often necessary to also learn long term dependencies of sequences. One solution to mitigate this problem is called "Gating". It is a mechanism to decide when to forget or remember a current input. By using gates, the network can learn which information it adds to or removes from the hidden state, resulting in the capability to learn long term dependencies. Well working and widely used Machine Learning Models, which make use of the "Gating" mechanism, are LSTM and GRU, which are the models we have used in our work and will present in the following.

5.1.2 LSTM

LSTM (Long Short Term Memory) networks are a special type of RNN, which mitigate the "Vanishing Gradient Problem" by using gates in order to learn long-term dependencies. LSTM networks were proposed by Hochreiter and Schmidhuber in 1997 and are very suitable for time series data and sequences [HS97]. They were successfully used in previous work [MK20], [JLZ+18] to recover a single gas concentration from the transient response.

In LSTM networks multiple cells are chained, like for RNNs. But the LSTM cells are more complex. They incorporates a cell state, an input gate, an output gate and a forget gate, which all have a different impact on the outputs of the cell. In Figure 5.2 one can

see the basic structure of an LSTM cell, which we will describe in detail in the following.

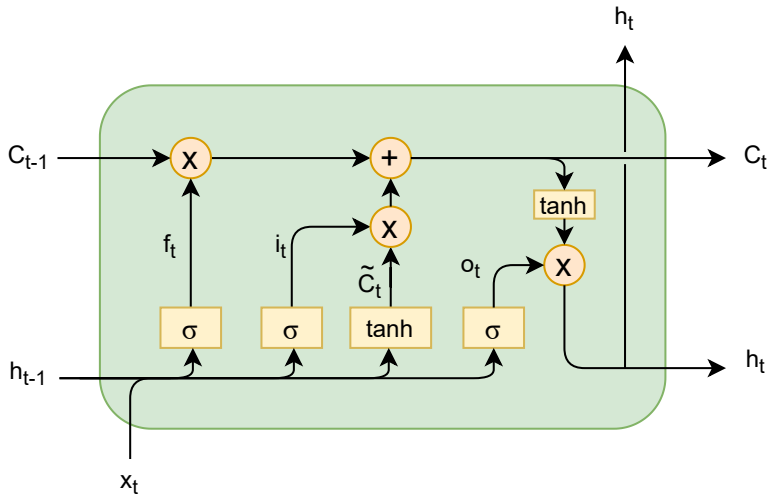


Figure 5.2: LSTM cell structure visualization. Drawing inspired by [Chr15].

The cell state C_t is a very important part of the LSTM cell. It contains information from previous LSTM cells and gives information to future LSTM cells. The three gates, which we already mentioned, control this information, the cell state, and decide, which parts of the cell state should be given to the next LSTM cell. A gate can be seen as a sigmoid layer, denoted with $\sigma(x) = \frac{1}{1+exp(-x)}$ representing the sigmoid function, which outputs values between zero and one, followed by a pointwise multiplication. It therefore can control how much of the parts and which parts of a vector can pass the gate or not. The inputs to the LSTM cell are the previous cell state C_{t-1}, the previous hidden state h_{t-1} and an input vector x_t. The forget gate layer takes x_t and h_{t-1} and decides, which parts of C_{t-1} it will keep, with

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{5.1}$$

Then, the input gate layer computes which parts of the cell state it will update, with

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{5.2}$$

New candidate values $\widetilde{C}_t$, which are added to the cell state depending on i_t are computed by

$$\widetilde{C}_t = tanh(W_C x_t + U_C h_{t-1} + b_C) \tag{5.3}$$

Having this, we can now create the current cell state C_t by taking into account the influences of the forget gate layer and input gate layer with

$$C_t = f_t \odot C_{t-1} + i_t \odot \widetilde{C}_t \tag{5.4}$$

The output h_t of the cell, which is also the new hidden state, is controlled by the output gate layer. The output gate controls which parts of a changed version of C_t will be output, giving h_t, with

$$\begin{aligned} o_t &= \sigma(W_o x_t + U_o h_{t-1} + b_o) \\ h_t &= o_t \odot tanh(C_t) \end{aligned} \tag{5.5}$$

In the equations, $W \in \mathbb{R}^{h \times d}$ and $U \in \mathbb{R}^{h \times h}$ denote matrices containing the weights which will be learned during training. $b \in \mathbb{R}^h$ denotes the bias vector which will be learned during training. d is the number of input features and h is the number of hidden units. The other variables represent vectors, like for example $x \in \mathbb{R}^d$, containing d values, depending on the number of input features. Further details can be found in [lst].

These days, LSTM networks are widely used, for example in speech recognition and time series prediction, and brought great success in many fields of artificial intelligence. There are also many variants of LSTM networks, which introduce changes to the LSTM cell structure with the goal to improve the overall performance. One very well known and successful variant is called GRU, which we will explain in the following.

5.1.3 GRU

GRU (Gated Recurrent Unit) is a variant of the LSTM, introduced by Cho et al. in 2014 [CvMG+14]. In comparison to LSTM, GRU has only two gates, an update gate and a reset gate and fewer parameters [Den15]. Additionally, the cell state is replaced by the hidden state only, which is also the output of each GRU cell. The inputs to the GRU cell are the previous cell state h_{t-1} and an input vector x_t. The reset gate layer takes x_t and h_{t-1} and decides how to merge the new input x_t and the previous output h_{t-1}, which contains the memory until now, with

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r) \tag{5.6}$$

Then, new candidate values $\widetilde{h}_t$ to add to the hidden state can be computed by

$$\widetilde{h}_t = tanh(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h) \tag{5.7}$$

The update gate decides, by using x_t and h_{t-1}, which parts of the candidate values $\widetilde{h}_t$ to add to form the new hidden state and output h_t and simultaneously changes the hidden state h_{t-1} with the opposite values from the update gate, reflected with

$$\begin{aligned} z_t &= \sigma(W_z x_t + U_z h_{t-1} + b_z) \\ h_t &= (1 - z_t) \odot h_{t-1} + z_t \odot \widetilde{h}_t \end{aligned} \tag{5.8}$$

In Figure 5.3 one can see the structure of a GRU cell.

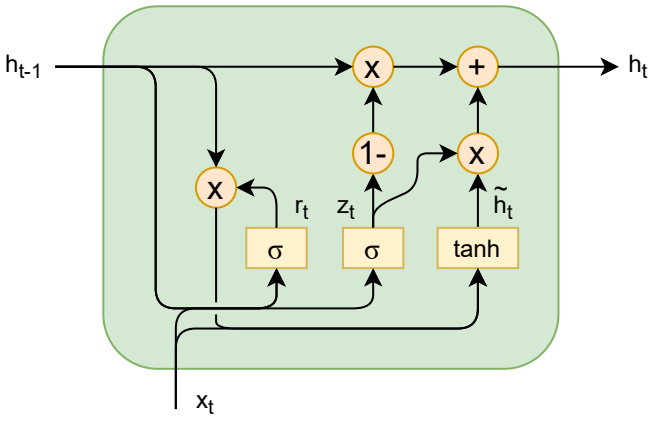


Figure 5.3: GRU cell structure visualization. Drawing inspired by [Chr15].

5.1.4 XGBoost

XGBoost, eXtreme Gradient Boosting, is an open-source software library, which allows to train Machine Learning Models using tree boosting and the gradient boosting framework. It was created as part of a research project at the University of Washington [xgb15], with a paper [CG16] presented at the SIGKDD Conference in 2016 [Vis19]. This method, which is based on decision trees, is widely used in the Machine Learning community and performing very well at Machine Learning competitions [xgb20].
The main idea behind XGBoost is, that various small models are created, added and combined, in order to minimize errors introduced by other models or model combinations. The ensemble of all models calculates the final predictions. [SN19]
The XGBoost implementation and documentation can be found in [xgb15] and [xd]. XGBoost offers the possibility to easily and quickly build models, which produce good results.

5.1.5 AutoML

Automated Machine Learning (AutoML) [aut] allows to automatically create working Machine Learning Models, by only providing raw data. Internally, this method automatically solves different tasks, like for example data preprocessing techniques, feature engineering, hyperparameter optimization and model selection, which usually have to be done by experts for usual machine learning models. AutoML allows non-experts to quickly and easily create models, which often provide better performance than Machine Learning Models designed by hand from scratch.
Although AutoML brings a lot of advantages, it is not suited for very complex problems, where experts are still needed. There are multiple companies offering AutoML solutions, like for example Microsoft Azure AutoML or Google Cloud AutoML, and also open source AutoML implementations, like for example from auto-sklearn, which we use in this work [SN20].
Auto-sklearn is an open source python library to create AutoML models, which was created by Matthias Feurer et al. and described in their paper in 2015 [FKE+15]. By using auto-sklearn, one not only finds the best working model among different possible models and configurations, but an ensemble from the best working models, which were found

during the process, is created, so that the overall performance increases [Jas20]. With this method, the general existing AutoML performance could be improved, as described in [FKE+15].

5.2 Data Preprocessing

In order to create Machine Learning Models with good performance, it is almost always necessary to apply preprocessing to the raw data, on which the models are trained. Different models have different requirements on their input data, in order to maximize their performance. Additionally, one can add robustness to the models or extract relevant information from the raw data with preprocessing, which otherwise would not have been visible. In our work, we have implemented three different data preprocessing techniques, which were used when needed while searching for the best performing models. The three techniques are:

(1) *Log Transformation.* Log transformation can be used to linearize data, whose underlying structure is exponential. The gas concentration in the environment and the corresponding sensor's raw signal change show this relationship. In the creation process of our models, we therefore consider log transformation as a data preprocessing technique which could improve performance.

(2) *Normalization.* Normalization is a well known data preprocessing technique and widely used in Machine Learning. The main idea of normalization is, that the input data is scaled and shifted to another range and the relationships between the values remain the same. Mostly, the input data is normalized to be in the range $[0, 1]$ or $[-1, 1]$. With normalization, one mitigates the problem that big values in samples have a higher influence on the model creation, because bigger values are seen as more important by the models. Further, many Machine Learning Models like for example LSTM networks or GRU networks benefit from the input values being in the previously mentioned ranges.

There are different forms of normalization. One can normalize each sample on its own using its respective maximum and minimum values or normalize the whole dataset using a "global" minimum and maximum value. Additionally, the minimum and maximum values can either be computed from the data, as described before, or can be fixed, because they are known due to knowledge of the data. In our work, we know practical minimum and maximum values for our sensor readings and therefore can use them to normalize the whole data with fixed minimum and maximum values. One can normalize a sensor raw value to a specified range, like for example $[-1, 1]$, by using

$$x_{norm} = \frac{x - d_{min}}{d_{max} - d_{min}}(r_{max} - r_{min}) + r_{min} \tag{5.9}$$

where:

x is the value to be normalized

x_{norm} is the normalized value

d_{max} is the maximum value for the data

d_{min} is the minimum value for the data

r_{max} is the maximum value for the range, *e.g.*, $r_{max} = 1$

r_{min} is the minimum value for the data, *e.g.*, $r_{min} = -1$

(3) *Data Augmentation.* Machine Learning Models usually need to be trained on a large amount of data, which is not always available. Data augmentation is a technique to artificially increase the amount of data available, in order to have enough data to train the model, to increase its robustness and to reduce overfitting. In our work, we use data augmentation to create additional slightly altered transient responses based on the real transient responses. We randomly shift them up and down by a small value, which should model the sensor's measurement uncertainties and noise. The models should be able to correctly interpret transient responses with small deviations, because in reality the sensors will never perfectly produce the same transient responses due to various unpredictable influences. By introducing data augmentation, we have managed to increase the performance of our models significantly.

5.3 Evaluation Metrics

An evaluation metric describes the performance of a model, by analyzing the differences between the ground truth values y_i, also known as targets, and the outputs of the model x_i, also known as predictions. In our work, we use Mean Absolute Error (MAE) for most of our analysis as an evaluation metric, but also provide Root Mean Square Error (RMSE) inside the plots.

Mean Absolute Error (MAE) computes the mean difference across targets and predictions and is easily interpretable. For some targets y_i and predictions x_i of length N, MAE is mathematically defined as

$$MAE = \frac{1}{N}\sum_{1}^{N}|y_i - x_i| \tag{5.10}$$

Root Mean Square Error (RMSE), also used in similar research [MK20], computes the square root of the mean of squared differences across targets y_i and predictions x_i. Larger differences have an higher impact on the error, which is why RMSE is sensitive to outliers. For some targets y_i and predictions x_i of length N, RMSE is mathematically defined as

$$RMSE = \sqrt{\frac{1}{N}\sum_{1}^{N}(y_i - x_i)^2} \tag{5.11}$$

Both, MAE and RMSE, are scale-dependent evaluation metrics. We therefore scale the ground truth values and predictions, which can be different for various models, to the same ranges before applying the error measures, so that we can compare the results obtained for different models.

Chapter 6

Experimental Work

Our experimental work is divided in two parts. In the first part, we analyze the behaviour of a sensor operated in *Duty Cycle Mode* and how to obtain reliable and accurate measurements through ML Models, which are comparable to measurements obtained from a sensor operated in *Continuous Mode*. In the second part, we investigate the behaviour of a sensor operated in *On Demand Mode* and its challenges compared to sensors operated in *Duty Cycle Mode*. We present our implemented approaches and their achieved performances. Because of the complexity of these problems, we focus our experiments solely on pixel 1 (P1) and its measurement values, raw signal Ethanol and tVOC. Nevertheless, a short analysis for pixel 2 (P2) is also presented, providing interesting insights into H2 and CO2-eq for future work.

During our experiments, we observed that the values obtained from all sensors are consistently higher than one would expect in an indoor environment for both tVOC and CO2-eq. Without further reference measurements we can neither confirm nor reject this claim, but similar values were consistently measured with five different SGP30 sensors. We also failed to identify a pollution source which may cause elevated pollution levels. We therefore trust the values provided by the SGP30 sensors and assume these are correct, while keeping in mind that the measurements may have an elevated baseline. This does not affect our findings and claims, yet the magnitude of the reported errors may be even lower than reported.

6.1 Predicting *Continuous Mode* Signal from *Duty Cycle Mode* Signal

Sensors operated in *Duty Cycle Mode* have a smaller sensitivity and suffer from surface contamination and other influences. We want to match the sensitivity of a sensor operated in *Duty Cycle Mode* to one operated in *Continuous Mode* and solve this problem by applying ML Models. As already described in Section 3.4, we assume that there is valuable information in the transient response, which can be used to estimate measurements close to ground truth measurements, which are provided by a sensor operated in *Continuous Mode*. We use $D_{train+test}$, where we have a sensor operated in *Duty Cycle Mode* with a period of 5 minutes and where on-times up to 5 seconds can be simulated. In order to find good working ML Models, we have to search for the best suitable data preprocessing

methods, described in Section 5.2, to apply to the raw data, the transient responses, and the best suitable hyperparameters for the models, described in 5.1, yielding the best possible performance. Therefore, we create various settings across our parameter space with which we search for the best performing model. The maximum length of one transient response for an on-time of 5 seconds is 180 measurement values, as we can measure with $36Hz$. We search for models working with different lengths of the transient responses, because it may be sufficient for models to look at shorter transient responses, which in practice would reduce the energy consumption even more. The length of the transient response used is denoted as W, as already described in Section 3.4. In general, the assumption is, that the longer the used transient response is, the better the performance of the model will be, as there is simply more information available. We have chosen LSTM, GRU, XGBoost and AutoML models, which are described in detail in Section 5.1.

The data is first split into a train (80%) and test (20%) set, on which the models are trained and evaluated respectively. A small part of the train set is used for validation, in order to reduce overfitting. To create an ML Model, we perform data preprocessing on the available data and use it to train various models based on the previously defined settings and hyper parameter combinations. After training, the model's performance is evaluated on the test set, which naturally was not used for training. We present the best results obtained on the test set for different model types and transient lengths W in Table 6.1, showing the performances expressed as Mean Absolute Error (MAE) in *ticks* and *ppb* between the ground truth and the predictions.

Model	*W*	*MAE [ticks]*	*MAE [ppb]*	*Model*	*W*	*MAE [ticks]*	*MAE [ppb]*
LSTM	40	178	223	XGBoost	40	235	356
LSTM	80	168	192	XGBoost	80	178	182
LSTM	160	133	154	XGBoost	160	158	160
GRU	40	184	234	AutoML	40	243	415
GRU	80	166	181	AutoML	80	194	243
GRU	160	136	143	AutoML	160	141	140

Table 6.1: P1. Performances of the best performing ML Models on the $D_{train+test}$ test set.

As one can observe for all ML models, the errors become lower the longer the transient response being used is, thus higher W positively impacts the models performances. This is expected, as longer transients contain more information which can be used by the models. One therefore has to make a trade-off between lower energy consumption and higher prediction errors because of shorter transient responses, and thus smaller W, used and higher energy consumptions for higher W, but with the advantage of getting better predictions. Further, one can observe that LSTM and GRU perform slightly better than XGBoost and AutoML. The reason for this could be, that LSTM and GRU work very well for time series and XGBoost and AutoML do not manage to learn the relationships between the values of a transient so well. LSTM and GRU have very similar performances. In the following, we will focus on the analysis of the GRU Models, especially for $W160$, as it was providing a very good performance. The analysis could be done analogously for the other models.

In Figure 6.1 we show the test set from $D_{train+test}$ and the predictions obtained by the best performing GRU model with $W160$. We can see in the plot, that the signal from the sensor operated in *Duty Cycle Mode* does not have prominent changes and stays mostly stable. Using the transient responses of this signal as an input for the model, we can calculate predictions which are near the ground truth values obtained by the sensor operated in *Continuous Mode*. We can conclude that there is enough information available in the transient responses to recover a signal which approximates the ground truth with satisfying accuracy.

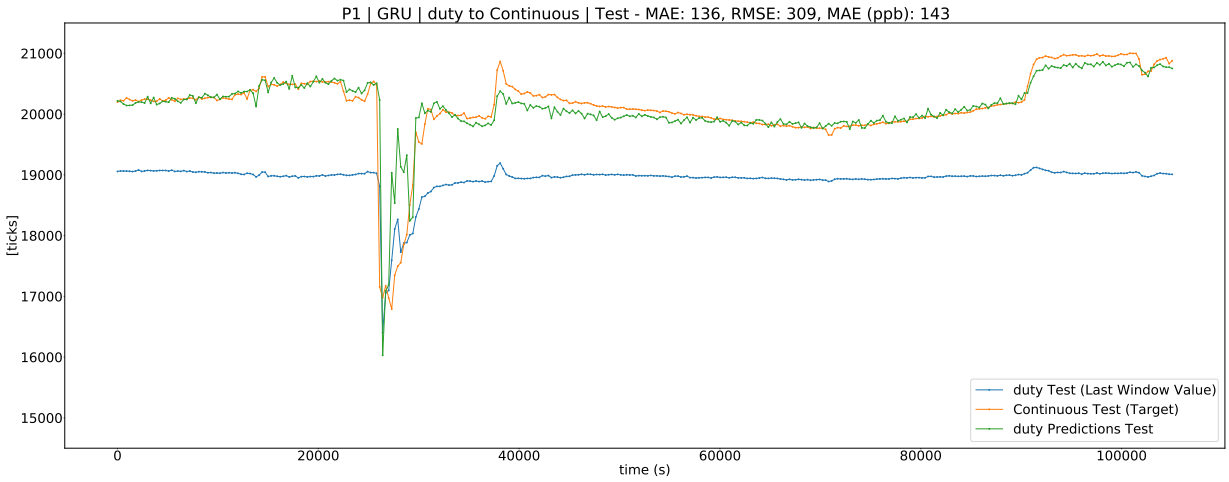


Figure 6.1: P1. Ground truth, *Duty Cycle Mode* signal and predictions generated with GRU, for $W160$ in *ticks* on $D_{train+test}$ test set. We can see predictions following the ground truth, while the *Duty Cycle Mode* signal used as input for the model is remaining mostly stable.

The question arises, whether the obtained performance is good enough for a real world application. For this, we have converted our signals to *ppb*, as described in Section 3.7, a unit which can be related to indoor air quality levels, described in Section 3.1.1. Additionally, we have shifted and scaled the *Duty Cycle Mode* signal for the plots using Linear Regression so that it fits the ground truth as good as possible in order to better see the advantages of the model. This is done, because in the field one would calibrate a sensor in *Duty Cycle Mode* in such a way, so that its output values are in reasonable ranges compared to a reference device. In Figure 6.2 we show the signals seen before in Figure 6.1, but converted to *ppb* in relation to indoor air quality levels.

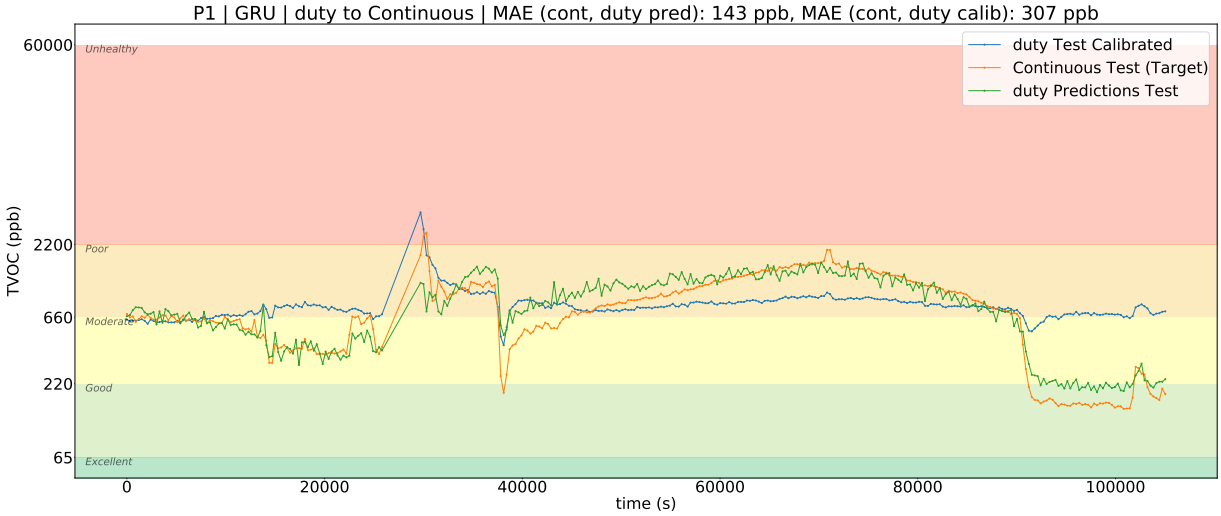


Figure 6.2: P1. Ground truth, *Duty Cycle Mode* signal and predictions generated with GRU, for $W160$ in *ppb* on $D_{train+test}$ test set. We can see predictions following the ground truth, while the *Duty Cycle Mode* signal used as input for the model is remaining mostly stable.

In Figure 6.2, we show the test set from $D_{train+test}$ in *ppb* ranging over different indoor air quality levels visualized with colors, from *Excellent* (green) to *Unhealthy* (red), which are described in Section 3.1.1. The *Duty Cycle Mode* signal remains nearly stable at around 660*ppb*, with some spikes induced by events. The ground truth signal, coming from the sensor operated in *Continuous Mode*, on the other hand, shows more prominent changes in air quality and also variation in *ppb* over the whole timespan. It is very satisfying to see that the predictions obtained by using a trained GRU Model based solely on the transient responses from the *Duty Cycle Mode* signal as an input, can compute a predicted signal, which follows the ground truth signal very well along the indoor air quality levels, although somehow noisy. In the first part of the signal, we can observe the ground truth and prediction going down, while the *Duty Cycle Mode* signal increases slightly or remains stable. In the middle part, we have a long part where the *Duty Cycle Mode* signal remains stable with a small increase and decrease, while the prediction manages to follow the ground truth. In the end, we have an improvement of the air quality, which is really well modelled by the prediction, although the *Duty Cycle Mode* signal remains on its original level. These are all clear indications that there is information in the transient responses, which can be used to predict indoor air quality level changes not only in terms of specific levels, but also in terms of *ppb* values.

Similar results are obtained by using other model types for training and prediction. The performances are listed in Table 6.1. Nevertheless, we also want to show an AutoML Model applied on the $D_{train+test}$ test set with $W160$ in *ppb* in Figure 6.3. We can see again that the predictions are very satisfying and the same conclusions as before can be drawn.

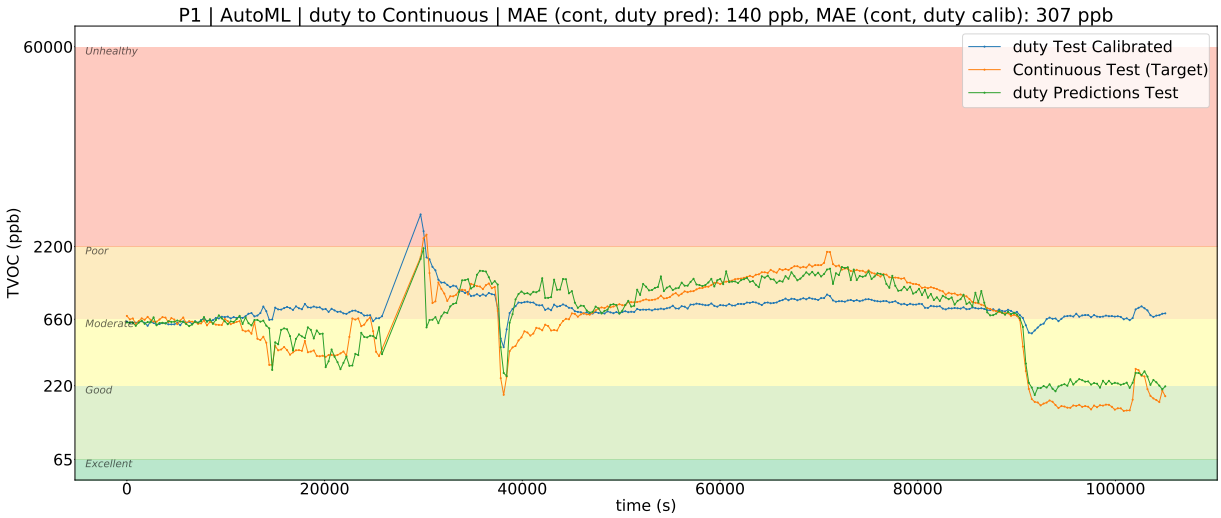


Figure 6.3: P1. Ground truth, *Duty Cycle Mode* signal and predictions generated with AutoML, for $W160$ in *ppb* on $D_{train+test}$ Test Set. We can see predictions following the ground truth, while the *Duty Cycle Mode* signal used as input for the model is remaining mostly stable.

Further, it is interesting to evaluate if the already trained models still work for data captured in the future. When do the models break and when do they have to be retrained? We use D_{test}^{+4d}, D_{test}^{+18d}, D_{test}^{+22d} and D_{test}^{+53d}, which are described in Section 4, for this analysis. The most important thing to notice about these datasets is that they were taken multiple days after the end of $D_{train+test}$ used for training. As already said, we focus the upcoming analysis on the best performing GRU Model, for $W160$, but also provide performances for the other models. We expect the other models to show the same behaviour, as future data is impacted by sensor drift which is independent from the models. The performances of the models applied on the datasets taken after training, for $W160$, can be seen in Table 6.2.

Model	*Dataset*	*W*	*MAE [ticks]*	*MAE [ppb]*
LSTM	D_{test}^{+4d}	160	241	320
LSTM	D_{test}^{+18d}	160	611	969
LSTM	D_{test}^{+22d}	160	787	771
LSTM	D_{test}^{+53d}	160	664	1348
GRU	D_{test}^{+4d}	160	227	256
GRU	D_{test}^{+18d}	160	499	880
GRU	D_{test}^{+22d}	160	592	676
GRU	D_{test}^{+53d}	160	652	1337
XGBoost	D_{test}^{+4d}	160	248	323
XGBoost	D_{test}^{+18d}	160	435	843
XGBoost	D_{test}^{+22d}	160	617	681
XGBoost	D_{test}^{+53d}	160	877	1533
AutoML	D_{test}^{+4d}	160	232	305
AutoML	D_{test}^{+18d}	160	473	845
AutoML	D_{test}^{+22d}	160	627	707
AutoML	D_{test}^{+53d}	160	751	1428

Table 6.2: P1. Pre-trained ML Models, $W160$, applied on data recorded after training.

By looking closer to Table 6.2, we can observe an increase of errors on data recorded farther away in time from the data which was used for model training. The model predictions get worse and worse for future data. This can happen because of the change in sensitivity of the sensor over time, where different sensor applications alter the sensitivity in different ways. It is also to be expected, that the sensor experiences different surface changes and a drift of measurement values over time. This leads to the conclusion, that either the models should be retrained after some time or the sensor drift should be modelled in some way, so that either the inputs to or the outputs from the models can be altered accordingly. While the pre-trained model has bad performance from D_{test}^{+18d} onwards, it still has a very good performance on D_{test}^{+4d}, a dataset not so far away in time from $D_{train+test}$. The performance of the GRU model, with $W160$, on this dataset can be seen in Figure 6.4. We can observe that although the data was recorded a few days after model creation, the predictions are consistent and still manage to follow the ground truth measurements and in most cases correctly predicting the indoor air quality levels. In the first part, we can observe higher misalignments, which come from the fact that the sensors need some time to settle in their mode.

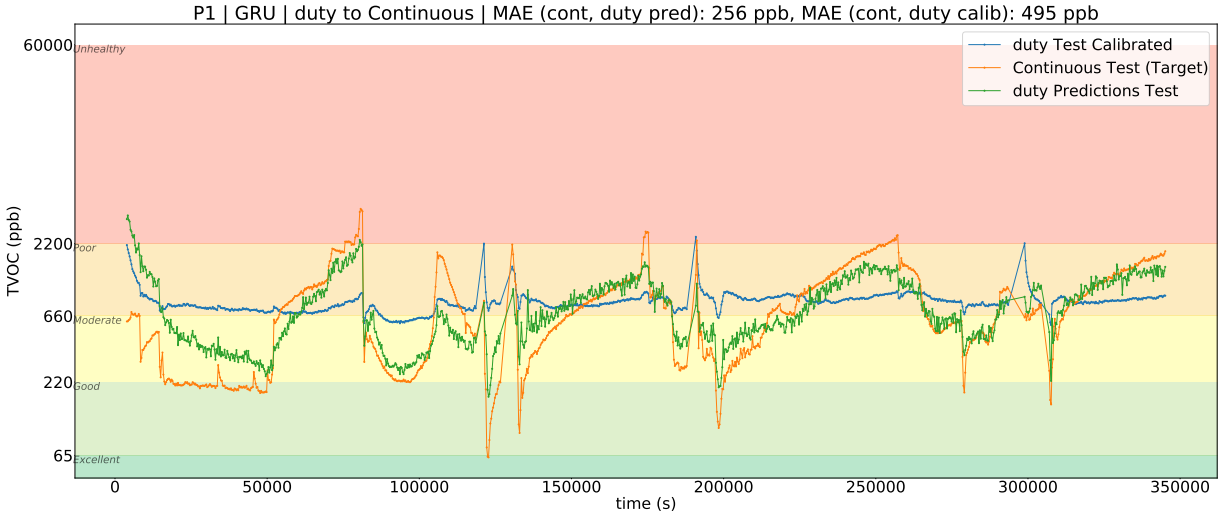


Figure 6.4: P1. Ground truth, *Duty Cycle Mode* signal and prediction from pre-trained GRU, $W160$, in *ppb* on D_{test}^{+4d}. Predictions are still following the ground truth very well.

While also examining the predictions for the other datasets, we have observed that the high errors do not come from completely random or unrealistic predictions. In fact, the predictions have a shape which is very similar to the ground truth, but are shifted away from it. We show this observation in Figure 6.5 for D_{test}^{+22d} and in Figure 6.6 for D_{test}^{+53d}.

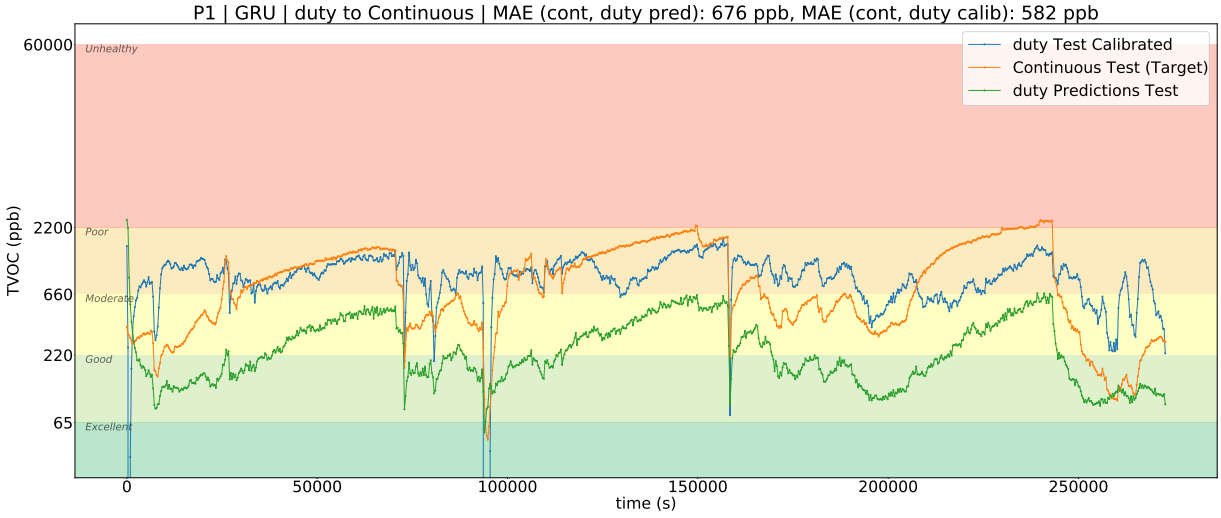


Figure 6.5: P1. Ground truth, *Duty Cycle Mode* signal and predictions from pre-trained GRU, $W160$, in *ppb* on D_{test}^{+22d}. We can observe well shaped predictions, but shifted from the ground truth.

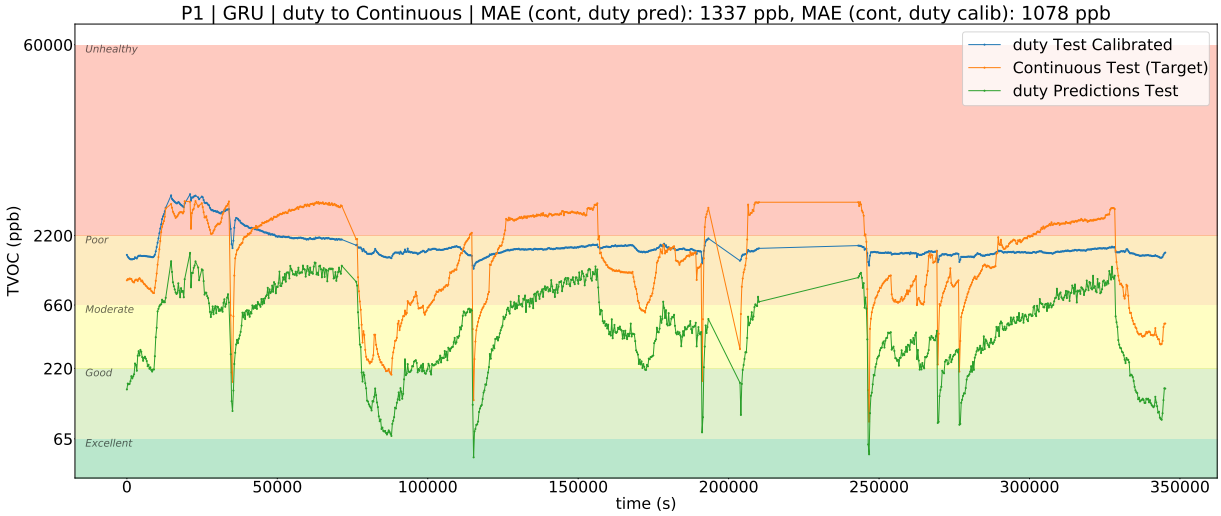


Figure 6.6: P1. Ground truth, *Duty Cycle Mode* signal and predictions from pre-trained GRU, $W160$, in *ppb* on D_{test}^{+53d}. We can observe well shaped predictions, but shifted from the ground truth.

We can conclude from these observations, that although a pre-trained model is used on future data, it still manages to provide reasonable predictions, but shifted. Nevertheless, one has to account for the shift of prediction in a post-processing step, in order to use pre-trained models for longer periods of time or maybe even account for a shift of the sensor operated in *Continuous Mode* and providing the ground truth measurements.

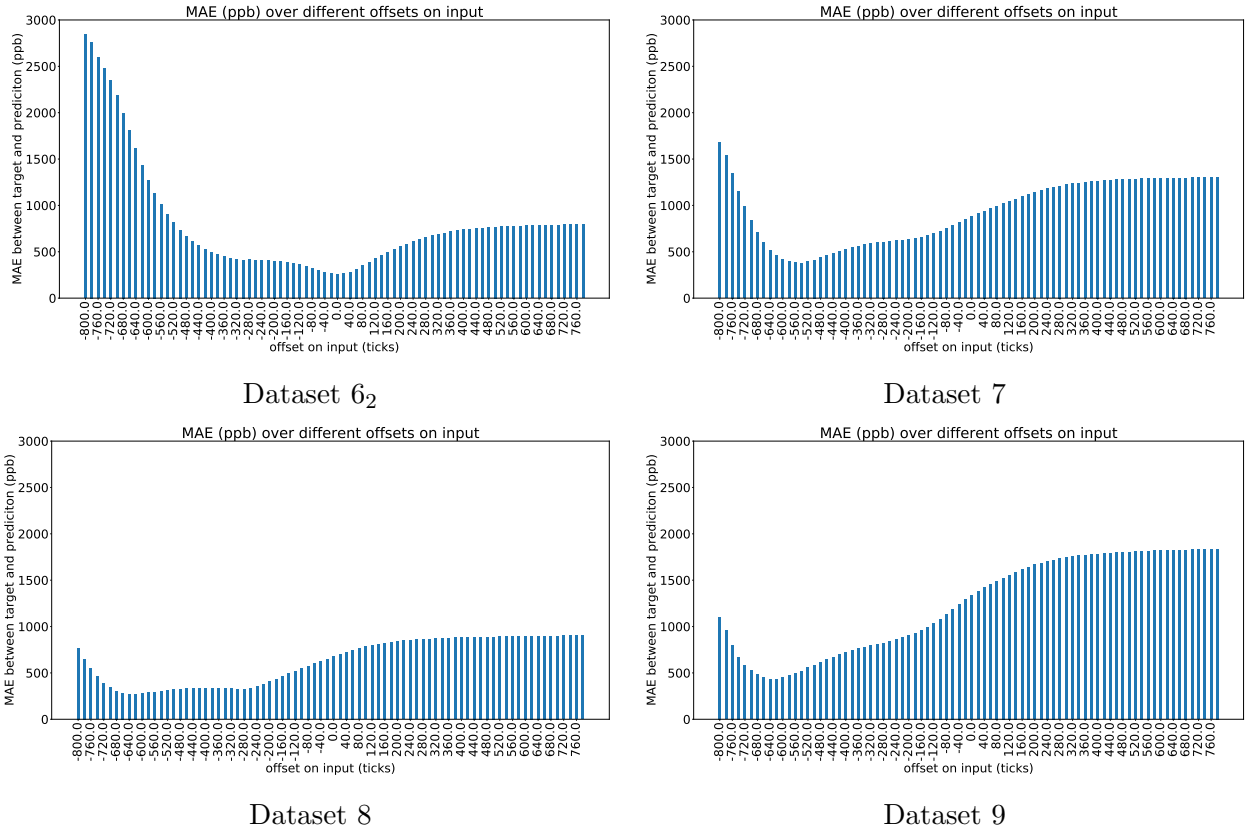
Another approach to minimize the impact of the previously mentioned effects, so that already trained models still work in an acceptable manner on future data, would be to add an offset to the input values for the models, which resulted in significantly improved performances. We have observed that for each dataset, there is a certain offset value, a so to say "sweet spot", where the performance of the pre-trained model is the best. As seen in Table 6.3, we can observe improved performances of the pre-trained model compared to Table 6.2 by applying the best possible offset on the input data before inference.

Model	Dataset	W	Offset [ticks]	*MAE [ticks]*	*MAE [ppb]*
GRU	6_2	160	20	221	262
GRU	7	160	-540	191	380
GRU	8	160	-280	191	328
GRU	9	160	-600	227	449

Table 6.3: P1. Pre-trained GRU Model, $W160$, applied on future data with applied offset before inference.

In Figure 6.7, one can see histograms of errors over different offsets, for the GRU Model with $W160$ applied on D_{test}^{+4d}, D_{test}^{+18d}, D_{test}^{+22d} and D_{test}^{+53d}. These histograms for

different offsets applied to the input data show very nicely, that even a simple method can already lead to improved performances with already trained models.



Dataset 6_2 Dataset 7

Dataset 8 Dataset 9

Figure 6.7: P1. Error histogram over different offsets applied on the input data for a pre-trained GRU model with $W160$, also showing the sweet spot indicating the offset for best performance.

By analysis of Table 6.3, we can conclude that the changes in sensitivity over time can be to some extent minimized by for example applying an offset to the input data so that good model performances are still obtained. Because of the fact that the errors can be described by a smooth and flat curve, especially around the sweet spot indicating the best performing offset, it is not necessary to exactly find the best offset value to get the best performance, but it is sufficient to find an offset value near the best possible one and still get a satisfying performance. In Figure 6.8 we show the performance of the model on D_{test}^{+22d}, where the best performing offset was applied before inference. The predictions are not shifted, but in the same range with respect to the ground truth. The predictions also manage to follow the ground truth well, considering the fact that the data was taken 22 days after model training.

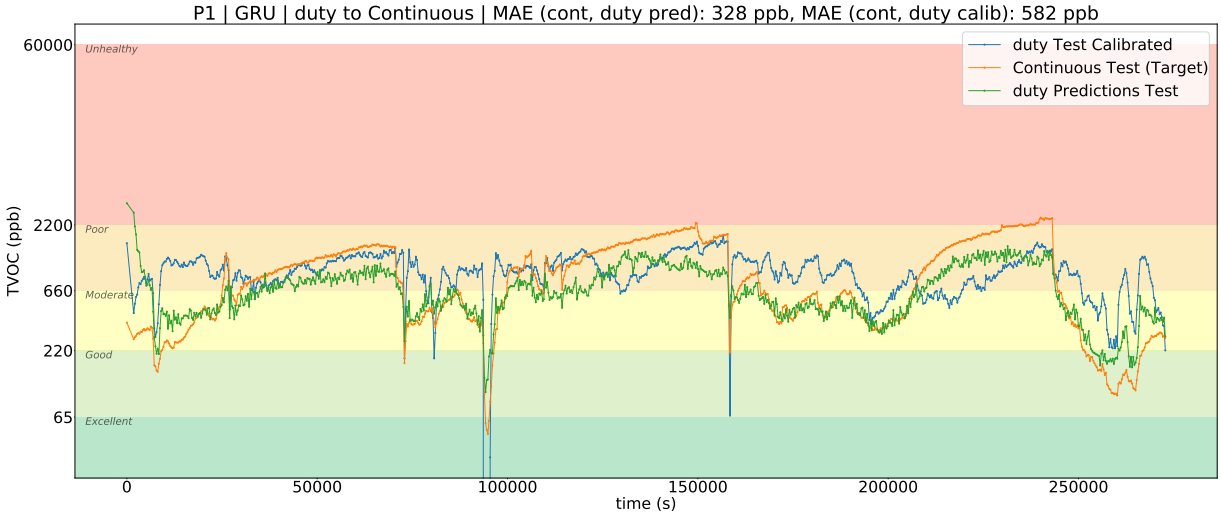


Figure 6.8: P1. Ground truth, *Duty Cycle Mode* signal and prediction generated with pre-trained GRU, for $W160$ in *ppb* on D_{test}^{+22d}. Offset applied to model input before inference.

Nevertheless, there still may be more sophisticated approaches to mitigate the mentioned sensor effects, which would need further research into the long term behaviour of the sensors. In conclusion we can see that over time, future data has to be in some way calibrated before inference or the predictions obtained based on future raw data have to be shifted and altered in some specific ways, so that the best possible performance for pre-trained model can be obtained. In addition, it may also be interesting to retrain models as new data is available, which on the other hand highly depends on the energy availability, the design and requirements of the application.

In conclusion, we could show that by using ML Models, one can get accurate and reliable measurements based on transient responses from a sensor operated in *Duty Cycle Mode*. We have shown that, from the transients alone, we could compensate for the change in sensitivity due to a different operation mode and for various possible effects influencing the measurements, demonstrating the presence of valuable properties in the transients. Moreover, influences by temperature and humidity and the cross-sensor offsets described in Section 3.6 could also be modelled by the ML models using only the transients. Further, we evaluated challenges which arise by using models over longer periods of time, which are expected and need to be taken into account for long-term reliability, but feasibility still could be shown.

We focused our experiments on pixel 1 and its tVOC measurements indicating indoor air quality, but we also wanted to provide a short analysis for pixel 2 and show that there is a big potential to repeat the previous experiments also for H2 and CO2-eq in the future. For this analysis, we have trained a GRU model, with $W160$, on H2 transient responses in *ticks* from $D_{train+test}$. We performed the same steps as described before, with data preprocessing, model training, model evaluation and conversion from *ticks* to *ppm*. In Figure 6.9 we show the performance of the best performing GRU model on the

$D_{train+test}$ test set, generating predictions from H2 transient responses obtained from the sensor operated in *Duty Cycle Mode*. The predictions manage to follow the ground truth well and CO2-eq levels are mostly correctly predicted.

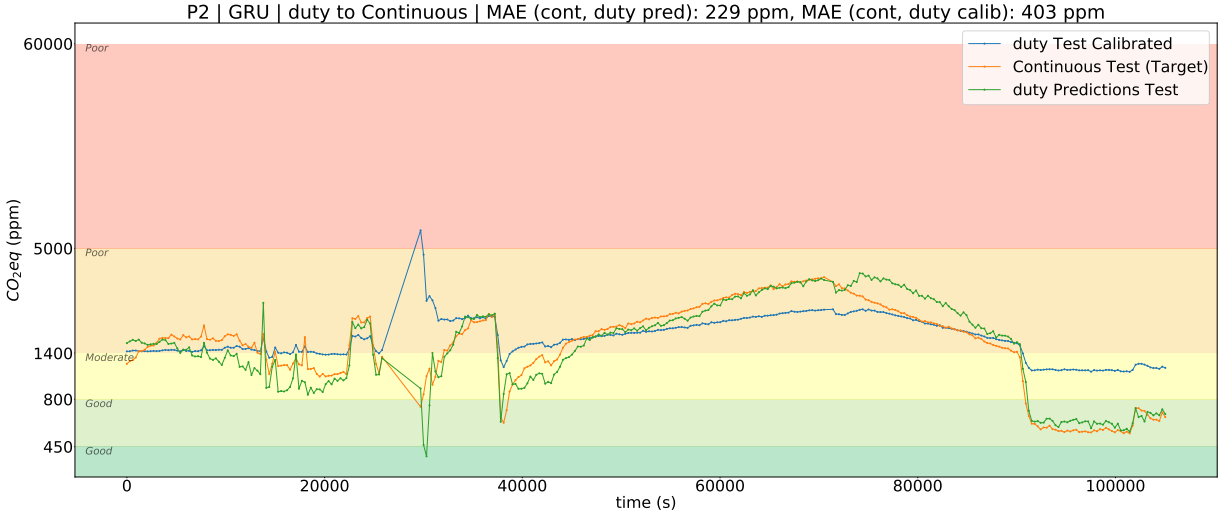


Figure 6.9: P2. Ground truth, *Duty Cycle Mode* signal and predictions from GRU, $W160$, in *ppm* on $D_{train+test}$ test set. Predictions show satisfying results for pixel 2.

We further applied this model on D_{test}^{+4d} to see whether the model also works on future data, which can be seen in Figure 6.10.

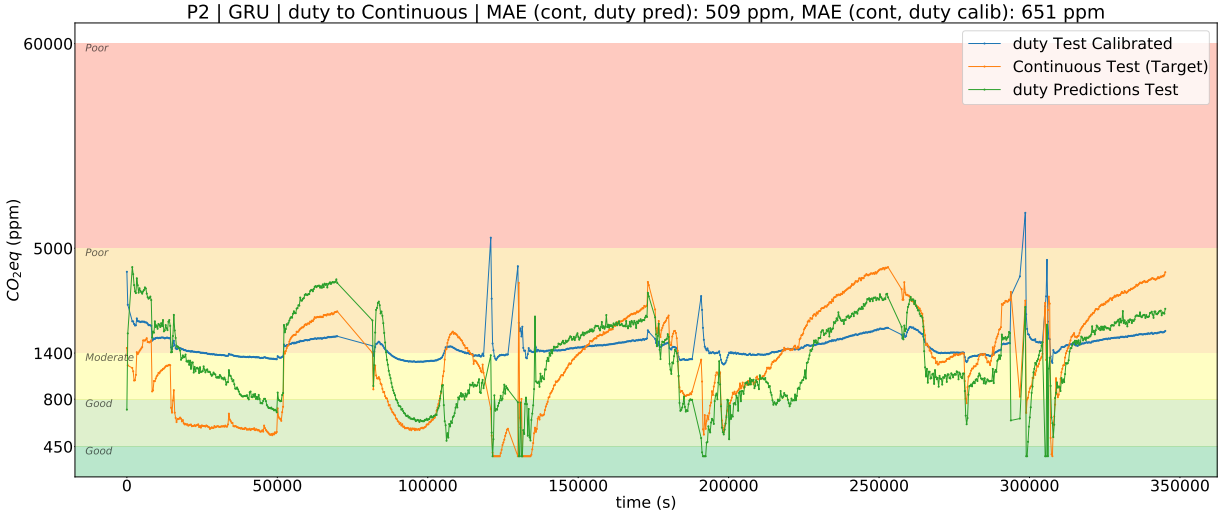


Figure 6.10: P2. Ground truth, *Duty Cycle Mode* signal and predictions from pre-trained GRU, $W160$, in *ppm* on D_{test}^{+4d}. Predictions are mostly reasonable, but improvable.

We can observe that the predictions of the model are quiet reasonable, but are not as good as the predictions obtained for pixel 1. Generally, we expect the challenges for pixel 2 to be more difficult than for pixel 1, as pixel 2 is not as sensitive to H2 as pixel 1 is to Ethanol. Further, the sensitivity of P2 to H2 even drops more when duty cycling, compared to P1, which is why it is more difficult to recover valuable information from a *Duty Cycle Mode* signal from P2. Considering these things, the models still manage to compute satisfying predictions also for pixel 2, which shows that it is worth investigating on how to create ML Models also for pixel 2 in order to obtain good measurements with low energy consumption.

6.2 *On Demand Mode* Analysis and Correction

As one can see in Chapter 4, the signals obtained by a sensor operated in *On Demand Mode* are spiky. Additionally, as seen in Section 3.4, the transient responses are very different to transient responses obtained from a sensor operated in *Duty Cycle Mode*. The reason for this is, that in *On Demand Mode*, we do not have constant off-times between the measurements, as we have in *Duty Cycle Mode*. These non constant off-times have an impact on the sensor. Longer off-times allow for example for more water to deposit on the sensor's surface which influences the measurements, as seen in the transient responses in Section 3.4. Further, chemical reactions could perform differently, when the previous measurement did not happen long ago and the hotplate target temperature is reached faster than usual. There may be also other complex effects which influence the transient responses of a sensor operated in *On Demand Mode*. Independent on the exact causes, we have observed that in *On Demand Mode*, the transient responses are influenced by the past off-times, leading to different transient responses even for the same air quality. Because of the fact, that *Duty Cycle Mode* and *On Demand Mode* are very similar, despite of the off-times changing for *On Demand Mode*, we would like to be able to use our trained models also for *On Demand Mode* data. But taking a good performing model from Section 6.1 to perform predictions based on *On Demand Mode* data, does not give good results, as the predictions are very spiky and do not allow for conclusions about the indoor air quality level.

We therefore investigate *On Demand Mode* measurements and how to minimize the occurring spikes, which are the result of measurement values being higher or lower due to different past off-times, in order to approach the signal obtained in *Duty Cycle Mode*, so that in future one could use the pre-trained models on *Duty Cycle Mode* data for *On Demand Mode* data. Towards this end, we analyze properties of *On Demand Mode* measurements based on data from $D_{train+test}$. We create small bands over the whole *Continuous Mode* signal (which is interpolated at the time steps of the *On Demand Mode* measurements), containing only values with approximately the same *ticks* value and thus values representing the same air quality. Each of these measurements has a corresponding *On Demand Mode* transient at the same time step. For each of these bands, we then plot different values from the *On Demand Mode* transients against the last off-time of the respective measurement. With this, we want to see how the last off-time impacts the measurement value under same air quality conditions. The plots for different W for one band can be seen in Figure 6.11.

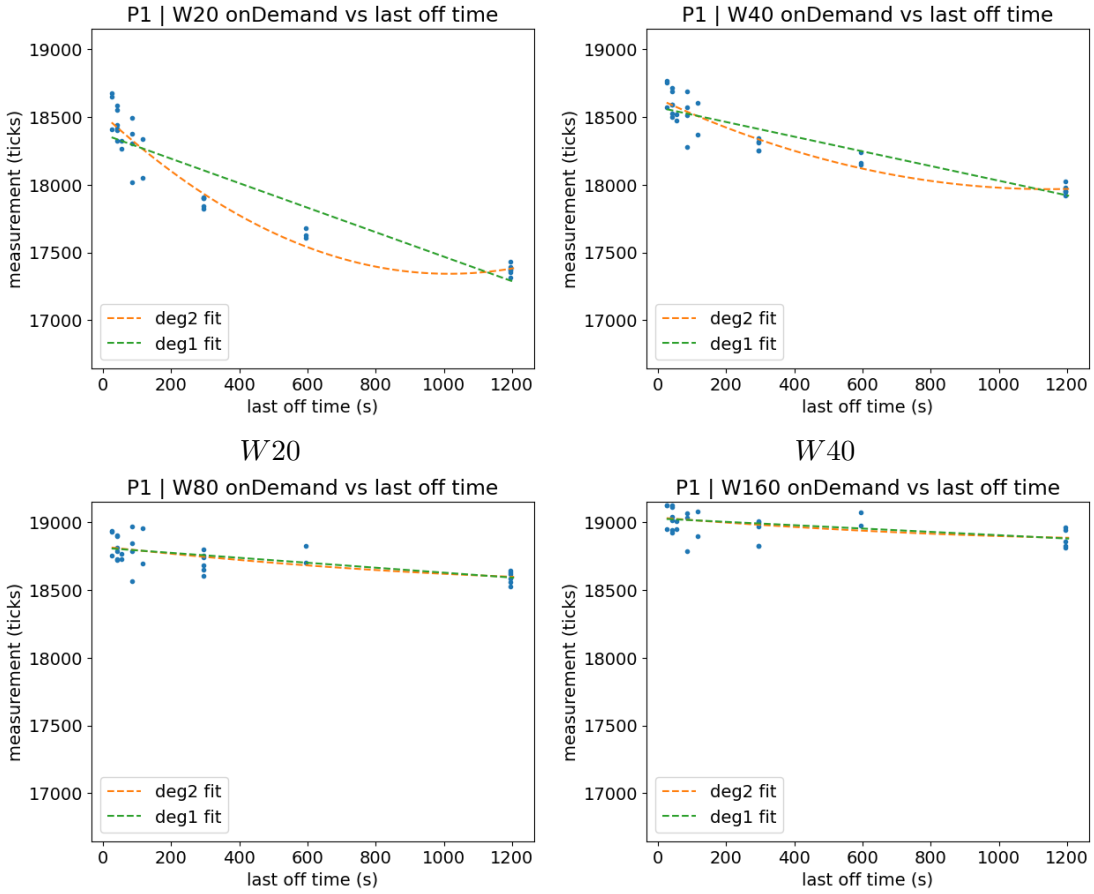
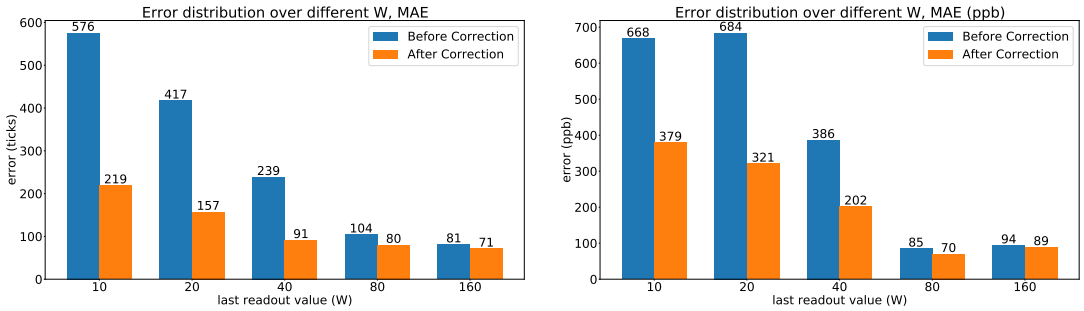


Figure 6.11: Visualization of values against their last off-time and their relationship in a specific band.

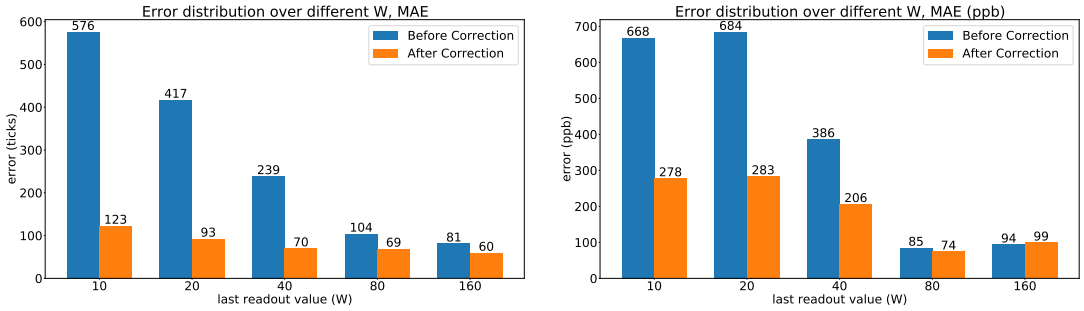
We have observed that for values underlying the same air quality, which is assured by the bands, we could fit a line based on their last off-times. This behaviour did hold for every W we have tested, namely $W10$, $W20$, $W40$, $W80$ and $W160$, with different slopes for the lines respectively, as seen in Figure 6.11. Although a 2-degree fit seems to fit the data points better, it did produce higher errors and more spiky results, which is why we are using the 1-degree fit for our correction. Using these findings, we can now create an algorithm to correct values for specific W. At first, we take the value and move it along the line with the respective slope to get d, the intersection of the line with the y-axis. After that we can now go from there back along the line to a pre-defined virtual off-time. For all values to be corrected, we choose the same virtual off-time, by which we simulate measurements in *Duty Cycle Mode*, which have the same off-time for every measurement. Using this approach, we could reduce the error between the *Duty Cycle Mode* and *On Demand Mode* signals as seen in Figure 6.12, especially for small W, where the effect is the strongest. This means, that high spikes of *On Demand Mode* measurements with small W can be minimized significantly.



MAE (*ticks*) MAE (*ppb*)

Figure 6.12: MAE (*ticks*) and MAE (*ppb*) histograms for different W, showing errors before and after correction of *On Demand Mode* values compared to *Duty Cycle Mode* values.

We further improved these results by averaging values over a time span of 20 minutes, which gave the best performance and also represents the highest possible off-time for *On Demand Mode*. The histogram of errors for different W can be seen in Figure 6.13.



MAE (*ticks*) MAE (*ppb*)

Figure 6.13: MAE (*ticks*) and MAE (*ppb*) histograms for different W, showing errors before and after correction, including averaging over time, of *On Demand Mode* values compared to *Duty Cycle Mode* values.

The impacts on the spikes after applying our proposed correction method and averaging over time is shown in Figure 6.14. We see that the *On Demand Mode* raw data with $W20$ is very spiky. The corrected signal on the other hand has significantly lower spikes and shows that an approximation to the *Duty Cycle Mode* signal should be feasible.

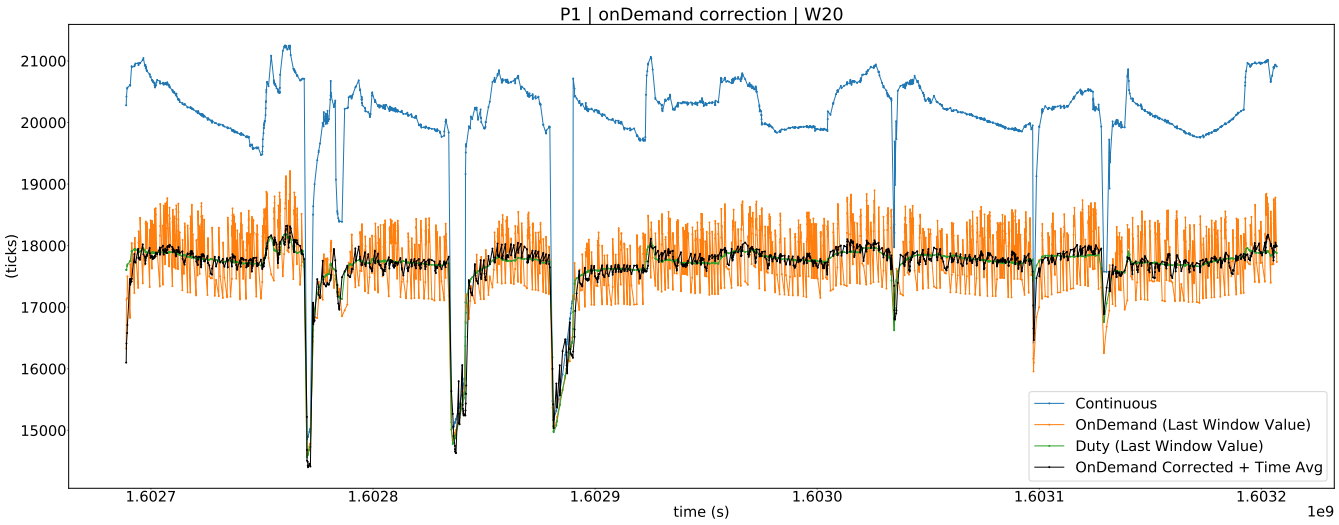


Figure 6.14: Correction of *On Demand Mode* signal from $D_{train+test}$, with $W20$. Spikes are significantly reduced.

We have shown that by analyzing the relationships between *On Demand Mode* transients and the last off-time, we were able to create a correction algorithm to minimize the difference between *On Demand Mode* and *Duty Cycle Mode* measurements. The effects were best visible for the earlier parts of the transient responses. For future work, we could imagine to evaluate the slopes of the lines for more W and to interpolate the missing ones, so that all points of the transients could be corrected based on a suitable line, with the goal that similar transients are obtained for measurements taken under the same air quality, like seen in *Duty Cycle Mode*. It is then to be examined, if the corrected transient responses mimic the behaviour seen for *Duty Cycle Mode* transient responses and in a further step, if the pre-trained models on *Duty Cycle Mode* data provide good predictions for the corrected *On Demand Mode* transients. Additionally, the impact of multiple previous off-times is also to be evaluated. As one can see, this topic imposes further analysis and measures to be taken, which is an interesting research field for our future work. Nevertheless, our findings serve as a good basis to solve the problems preventing models to predict accurate measurements from *On Demand Mode*. Due to our findings, we can suggest for a sensing-friendly scheduling when running on intermittent power. When operating the sensor on-demand, one should aim for constant off-times between the measurements, as this prevents challenges coming with irregular off-times. It is therefore beneficial to mimic a duty cycled sensor with scheduling and to measure in regular intervals, instead of directly measuring as soon as enough energy is available.

Chapter 7

Conclusion & Future Work

In this thesis, we show feasibility that by using Machine Learning Models, it is possible to predict accurate measurements from the transient responses obtained from a SGP30 gas sensor, which is duty-cycled with an on-time of under 5 seconds and an off-time of 295 seconds. This allows for energy savings of 98% compared to a sensor operating continuously, by introducing only a Mean Absolute Error of 143*ppb* for tVOC and by correctly predicting 80% of indoor air quality levels. Further, we have shown that such models also work for CO2-eq. We have presented details about the SGP30 sensor and its functionality, about challenges to be solved, about the design of Machine Learning Models and their application on data taken days or even weeks after model training. Based on our work we can say, that there is valuable information available in a transient response to recover an accurate measurement and to compensate for effects causing changes in sensitivity by using ML Models.

Moreover, we have shown that irregular operation of the SGP30 sensor with small on-times due to intermittent energy availability imposes additional difficulties which result in spiky signals. We have discovered a dependency of single measurement values with respect to their preceding off-time and proposed a correction algorithm to mitigate occurring spikes, where the best results were obtained for earlier parts of the transient responses. We also tried to predict accurate measurements directly from *On Demand Mode* signals, but did not manage to achieve good performances with various models due to the spikiness of the *On Demand Mode* data, which is why we conclude that it is important to first correct the *On Demand Mode* signal to a suitable virtual *Duty Cycle Mode* signal before inference.

Due to our analysis, we can suggest for applications running on intermittent power that one implements a scheduling, which does not schedule the measuring step as soon as energy is available, but schedules it with the perspective that model performance is increased when measurements are performed in regular intervals, as observed in *Duty Cycle Mode*.

As this topic is very interesting and promises to gain the attention of many, who want to operate gas sensors in low energy applications, we would like to present some possible additional improvements and motivation for future work. For a better understanding of the sensor's properties while duty cycling, it would be interesting to repeat our experiments with different periods. Also it may be useful to incorporate temperature and humidity as features for the ML Models, given that sensors providing these values are available. Further efforts can be invested in the evaluation and creation of models for pixel 2 in

order to predict accurate measurements for CO2-eq, as we have shown that it indeed is possible. While significantly lowering energy consumption by duty cycling the sensor, it is also important to evaluate the additional energy consumption for model inference and, if models are running on a continuously powered ground station, for data transmission and other relevant overhead. Moreover, we think that it would bring huge benefits to further analyze gas sensor's on-demand behaviour and to further improve the correction algorithm and create models, as this would really allow for operation on intermittent energy. This is also affected by implementing a sensing-friendly scheduling design, which can help to mitigate unwanted effects of irregular on-demand operation.

While the analysis of gas sensors from other companies might bring additional insights, there is one specific sensor which is very closely related to our work. Sensirion recently released the SGP40 gas sensor [Sen20d], the successor of the SGP30 used in this thesis, which promises far lower energy consumptions for tVOC measurements. An analysis of the sensors' differences, characteristics and duty cycling behaviour of the SGP40 in combination with the creation of new ML Models to even further reduce energy consumption, could really present a big milestone for gas sensors to be used in mobile or IoT devices, with very low and acceptable energy consumptions, allowing to introduce air quality monitoring nearly everywhere and to improve the lives of everyone.

Bibliography

[20220] NotAnotherOne Inc. 2020. Standards for indoor air quality (iaq). `https://help.atmotube.com/faq/5-iaq-standards/`, 2020. [Online; accessed 16-January-2021].

[ada20a] adafruit. Adafruit circuitpython busdevice. `https://github.com/adafruit/Adafruit_CircuitPython_BusDevice`, 2020. [Online; accessed 16-January-2021].

[ada20b] adafruit. Adafruit circuitpython sgp30. `https://github.com/adafruit/Adafruit_CircuitPython_SGP30`, 2020. [Online; accessed 16-January-2021].

[ALG+15] G. Adam, S. Lemaigre, X. Goux, P. Delfosse, and A.-C. Romain. Upscaling of an electronic nose for completely stirred tank reactor stability monitoring from pilot-scale to real-scale agricultural co-digestion biogas plant. *Bioresour. Technol.*, 178:285–296, 2015.

[aut] Automated machine learning. Wikipedia, `https://en.wikipedia.org/wiki/Automated_machine_learning`. [Online; accessed 16-January-2021].

[BFB+16] A. Brown, P. Franken, S. Bonner, N. Dolezal, and J. Moross. Safecast: Successful citizen-science for radiation measurement and communication after Fukushima. *Journal of Radiological Protection*, 36:82–101, 2016.

[BM18] J. Burgues and S. Marco. Low power operation of temperature-modulated metal oxide semiconductor gas sensors. *Sensors*, 18:339, 2018.

[CG16] Tianqi Chen and Carlos Guestrin. Xgboost. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Aug 2016.

[Chr15] Olah Christopher. Understanding lstm networks. `https://colah.github.io/posts/2015-08-Understanding-LSTMs/`, 2015. [Online; accessed 16-January-2021].

[co2] Indoor air quality. Wikipedia, `https://en.wikipedia.org/wiki/Indoor_air_quality#Carbon_dioxide`. [Online; accessed 16-January-2021].

[CvMG+14] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase

representations using rnn encoder-decoder for statistical machine translation, 2014.

[Den15] Britz Denny. Recurrent neural network tutorial, part 4 implementing a gru/lstm rnn with python and theano. `http://www.wildml.com/2015/10/recurrent-neural-network-tutorial-part-4-implementing-a-grulstm-rnn-with-python-and-theano/`, 2015. [Online; accessed 16-January-2021].

[dut] Duty cycle. Wikipedia, `https://en.wikipedia.org/wiki/Duty_cycle`. [Online; accessed 16-January-2021].

[FCAB10] G. Fine, L. Cavanagh, A. Afonja, and R. Binions. Metal oxide semi-conductor gas sensors in environmental monitoring. *Sensors*, 10(6):5469–5502, 2010.

[FKE+15] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. Efficient and robust automated machine learning. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28, pages 2962–2970. Curran Associates, Inc., 2015.

[FOU20] RASPBERRY PI FOUNDATION. Raspberry pi 3 model b. `https://www.raspberrypi.org/products/raspberry-pi-3-model-b/`, 2020. [Online; accessed 16-January-2021].

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[HSW+15] D. Hasenfratz, O. Saukh, C. Walser, C. Hueglin, M. Fierz, T. Arn, and L. Thiele. Deriving high-resolution urban air pollution maps using mobile sensor nodes. *IEEE Journal of Pervasive and Mobile Computing*, 16:268–285, 2015.

[Jas20] Brownlee Jason. Auto-sklearn for automated machine learning in python. `https://machinelearningmastery.com/auto-sklearn-for-automated-machine-learning-in-python/`, 2020. [Online; accessed 16-January-2021].

[JLZ+18] Zhenhua Jia, Xinmeng Lyu, Wuyang Zhang, Richard P. Martin, Richard E. Howard, and Yanyong Zhang. Continuous low-power ammonia monitoring using long short-term memory neural networks. In *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems*, SenSys '18, page 224236, New York, NY, USA, 2018. Association for Computing Machinery.

[KHK+08] Y. Kim, I. Hwang, S. Kim, C. Lee, and J. Lee. Cuo nanowire gas sensors for air quality control in automotive cabin. *Elsevier Sensors and Actuators B: Chemical*, 135:298–303, 2008.

[KLH+17] S. Khalifa, G. Lan, M. Hassan, A. Seneviratne, and S. Das. Harke: Human activity recognition from kinetic energy harvesting data in wearable devices. *IEEE Transactions on Mobile Computing*, PP:1–1, 10 2017.

[KPR+06] M. Kuske, Marta Padilla, Anne-Claude Romain, Jacques Nicolas, Ramón Rubio, and Santiago Marco. Detection of diverse mould species growing on building materials by gas sensor arrays and pattern recognition. *Elsevier Sensors and Actuators B: Chemical*, 119:33–40, 2006.

[KS16] H. Kalantarian and M. Sarrafzadeh. Pedometers without batteries: An energy harvesting shoe. *IEEE Sensors Journal*, 16(23):8314–8321, 2016.

[LCM+15] A. Loutfi, S. Coradeschi, G. Kumar Mani, P. Shankar, and J. Bosco. Electronic noses for food quality: A review. In *J. Food Eng.*, volume 144, pages 103–111, 2015.

[lst] Long short-term memory. Wikipedia, `https://en.wikipedia.org/wiki/Long_short-term_memory#LSTM_with_a_forget_gate`. [Online; accessed 16-January-2021].

[LXM+19] G. Lan, W. Xu, D. Ma, S. Khalifa, M. Hassan, and W. Hu. Entrans: Leveraging kinetic energy harvesting signal for transportation mode detection. *IEEE Transactions on Intelligent Transportation Systems*, PP:1–12, 2019.

[Mic18] Phi Michael. Illustrated guide to recurrent neural networks. `https://towardsdatascience.com/illustrated-guide-to-recurrent-neural-networks-79e5eb8049c9`, 2018. [Online; accessed 16-January-2021].

[MPS+13] M. Mead, O. Popoola, G. Stewart, P. Landshoff, M. Calleja, M. Hayes, J. Baldoví, M. McLeod, T. Hodgson, J. Dicks, A. Lewis, J. Cohen, R. Baron, J. Saffell, and R. Jones. The use of electrochemical sensors for monitoring urban air quality in low-cost, high-density networks. *Atmospheric Environment*, 70:186–203, 2013.

[ON 12] ON Semiconductor. *BC546B, BC547A, B, C, BC548B, C - Amplifier Transistors - NPN Silicon*, 2012. Available at `https://www.onsemi.com/pub/Collateral/BC546-D.PDF`, Document number 98AON52857E.

[PJDP17] Kirsty H. Grant Alex G. Brundle Martin R. Thompson Joshua Vande Hey R. R. Leigh Philip J. D. Peterson, Amrita Aujla. Practical use of metal oxide semiconductor gas sensors for measuring nitrogen dioxide and ozone in urban environments. *Sensors*, 17(7):1653, 2017.

[PPM+10] M. Padilla, A. Perera, I. Montoliu, A. Chaudry, K. Persaud, and S. Marco. Drift compensation of gas sensor array data by orthogonal signal correction. *Chemometrics and Intelligent Laboratory Systems*, 100:28–35, 2010.

[RAP15] M. Righettoni, A. Amann, and S. E Pratsinis. Breath analysis by nanostructured metal oxides as chemo-resistive gas sensors. *Materialstoday*, 18:163–171, 2015.

[RBL+17] A. Rodriguez, D. Balsamo, Z. Luo, S. P. Beeby, G. V. Merrett, and A. S. Weddell. Intermittently-powered energy harvesting step counter for fitness tracking. In *2017 IEEE Sensors Applications Symposium (SAS)*, pages 1–6, 2017.

[RHB18] Daniel Rffer, Felix Hoehne, and Johannes Buehler. New digital metal-oxide (mox) sensor platform. *Sensors (Basel, Switzerland)*, 18, 03 2018.

[RN09] A. Romain and J. Nicolas. Long Term Stability Of Metal Oxide-Based Gas Sensors For E-nose Environmental Applications: an overview. In *American Institute of Physics Conference*, volume 1137, pages 443–445, 2009.

[SBM+09] P. Sekhar, E. Brosha, R. Mukundan, M. Nelson, and F. Garzon. Application of commercial automotive sensor manufacturing methods for NO. In *ECS Transactions*. ECS, 2009.

[Sen19] Sensirion. *SVM30 - Multi-gas, humidity and temperature sensor combo module*, 2019. Available at `https://www.sensirion.com/fileadmin/user_upload/customers/sensirion/Dokumente/9_Gas_Sensors/Datasheets/Sensirion_Gas_Sensors_Datasheet_SVM30.pdf`, Version 1.1.

[Sen20a] Sensirion. General topic: Cmosens technology. `https://www.sensirion.com/en/about-us/company/technology/cmosens-technology-in-gas-sensing/`, 2020. [Online; accessed 16-January-2021].

[Sen20b] Sensirion. Sensirion controlcenter. `https://www.sensirion.com/de/controlcenter/`, 2020. [Online; accessed 16-January-2021].

[Sen20c] Sensirion. *SGP30 - Indoor Air Quality Sensor for TVOC and CO2eq Measurements*, 2020. Available at `https://www.sensirion.com/fileadmin/user_upload/customers/sensirion/Dokumente/9_Gas_Sensors/Datasheets/Sensirion_Gas_Sensors_Datasheet_SGP30.pdf`, Version 1.0.

[Sen20d] Sensirion. *SGP40 - Indoor Air Quality Sensor for VOC Measurements*, 2020. Available at `https://www.sensirion.com/fileadmin/user_upload/customers/sensirion/Dokumente/9_Gas_Sensors/Datasheets/Sensirion_Gas_Sensors_Datasheet_SGP40.pdf`, Version 1.0.

[Sen20e] Sensirion. *SGPC3 - Indoor Air Quality Sensor for TVOC*, 2020. Available at `https://www.sensirion.com/fileadmin/user_upload/customers/sensirion/Dokumente/9_Gas_Sensors/Datasheets/Sensirion_Gas_Sensors_Datasheet_SGPC3.pdf`, Version 1.0.

[Sen20f] Sensirion. Unique performance thanks to moxsens technology. `https://www.sensirion.com/en/environmental-sensors/gas-sensors/`, 2020. [Online; accessed 16-January-2021].

[SGL+16] Lukas Sigrist, Andres Gomez, Roman Lim, Stefan Lippuner, Matthias Leubin, and Lothar Thiele. Rocketlogger - mobile power logger for prototyping iot devices. In *14th ACM Conference on Embedded Networked Sensor Systems (SenSys 2016)*, Stanford, CA, USA, Nov 2016.

[SN19] Luber Stefan and Litzel Nico. Was ist xgboost? `https://www.bigdata-insider.de/was-ist-xgboost-a-844791/`, 2019. [Online; accessed 16-January-2021].

[SN20] Luber Stefan and Litzel Nico. Was ist automatisiertes machine learning (automl)? `https://www.bigdata-insider.de/was-ist-automatisiertes-machine-learning-automl-a-896975/`, 2020. [Online; accessed 16-January-2021].

[Vis19] Morde Vishal. Xgboost algorithm: Long may she reign! `https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d`, 2019. [Online; accessed 16-January-2021].

[WB11] A. Wilson and M. Baietto. Advances in electronic-nose technologies developed for biomedical applications. *Sensors (Basel)*, 11:1105–76, 2011.

[WBF+12] J. Wächter, A. Babeyko, J. Fleischer, R. Hner, M. Hammitzsch, A. Kloth, and M. Lendholt. Development of tsunami early warning systems and future challenges. *Natural Hazards and Earth System Sciences*, 12:1923–1935, 2012.

[WYZ+10] C. Wang, L. Yin, L. Zhang, D. Xiang, and R. Gao. Metal oxide gas sensors: Sensitivity and influencing factors. *Sensors*, 10(3):2088–2106, 2010.

[xd] xgboost developers. Xgboost documentation. `https://xgboost.readthedocs.io/en/latest/`. [Online; accessed 16-January-2021].

[xgb15] Xgboost extreme gradient boosting. `https://github.com/dmlc/xgboost`, 2015. [Online; accessed 16-January-2021].

[xgb20] Awesome xgboost - machine learning challenge winning solutions. `https://github.com/dmlc/xgboost/tree/master/demo#machine-learning-challenge-winning-solutions`, 2020. [Online; accessed 16-January-2021].

[MK20] J. uli Gambiroa, T. Masteli, T. Kovaevi, and M. agalj. Predicting low-cost gas sensor readings from transients using long short-term memory neural networks. *IEEE Internet of Things Journal*, 7(9):8451–8461, 2020.