



Norwegian University  
of Life Sciences

**Master Thesis 2018 30 ECTS**

Faculty of Science and Technology  
Odd Ivar Lekang

# **CMUT based chemical sensor for classification and quantification with machine learning in a real- world application.**

**Maureen Byrne**

Machine, Process and Energy Technology  
Faculty of Science and Technology





# PREFACE

The purpose of this thesis is to demonstrate a proof of concept of a CMUT based chemical sensor as a gas detecting unit, that can classify and quantify chemicals with machine learning in a real-world application. The thesis is a continuation of research done at Stanford University on the CMUT chemical sensor, funded by Fluenta. Data this thesis is built on was collected during winter 2017. Collecting the data was repetitive and time consuming, thus the development of a robot was suggested. The suggestion was taken seriously which led to the design of an autosampler during the course of this thesis. Proving itself as an invaluable tool which hopefully will increase the efficiency of the data collection and help build a data library for the sensor which future machine learning models can be trained on.

I am thankful to *Eik Idéverksted* for the opportunity to partake in the research on the CMUT sensor. It has been rewarding and fulfilling, adding to my five years of study. Especially learning the methods of machine learning from scratch and successfully employing models on the data from the field. A special thanks to Ola and Kristian Omberg whom have been excellent consultants for academic feedback and motivation throughout the thesis work. I would also like to thank Magnus R. Nyvold for proof reading and my fellow students for support and entertainment during the semester. Last but not least a special thanks to my family for love and support.

Ås

May 15<sup>th</sup>, 2018

---

*Maureen Byrne*





## ABSTRACT

In a quest for further enhancing human senses, chemical sensors are developed. Chemical sensors are proved to diagnose diseases, classify and quantify chemical warfare agents as well as measuring air pollution down to parts per billion [1-3]. Connecting multiple devices in large networks can help authorities and governments respond faster and make better decisions considering release of emissions and/or dangerous gases. In order to create such networks, a inexpensive, robust and portable sensor must be developed. The chemical capacitive micromachined ultrasonic transducer (CMUT) might be such a sensor.

This thesis demonstrates a proof of concept for a CMUT based chemical sensor as a gas detecting unit that can classify and quantify chemicals with machine learning in a real-world application. The CMUT is a sensor consisting of an array of polymer coated cells adsorbing different gases. Adsorption causes a frequency shift in the sensor output. This shift can be correlated to chemicals and their concentrations through machine learning. Reference data collected for the machine learning models was identified as a time-consuming process. An autosampler was devised, reducing time and cost related to the data collection. The CMUT sensor was tested in a greenhouse for 4 weeks to measure CO<sub>2</sub> concentration in a plant bed under varying conditions. Testing the following statement: If the sensor can detect low concentrations of CO<sub>2</sub> in ambient air it can also detect other compounds. The machine learning models were trained on the collected samples, and later compared to find the best model.

The results showed that the CMUT sensor successfully measured CO<sub>2</sub> down to 120 ppm in ambient air, the machine learning models could classify between high and low concentrations. For classification purposes the *neural network with relu activation* showed the best results, with a *15% error for both high and low concentrations*. Quantification of the data had poor performance due to sensor drift. *Large RMSE scores was found for all quantification models*. The drift is most likely caused by the breakdown of the polymer, causing a frequency shift. The dataset was unbalanced and had a higher distribution on lower concentrations. Which to some extent undermine the results from the machine learning, although giving an indication of sensor performance. Further research is recommended to assess the polymer coating on the CMUT as well as removing drift. Reducing the size of the sensor and equipment, as well as connecting the sensor to a cloud database, is recommended and identified as important steps for creating a sensor network.



# SAMMENDRAG

I søken etter å forbedre menneskets sanser ønsker man å utvikle kjemiske sensorer. Kjemiske sensorer har blitt brukt til å diagnostisere sykdommer, klassifisere og kvantifisere nervegass i tillegg til å måle luftforurensning som har svært lav oppløsning. Ved å sette sammen flere elektroniske neser i større nettverk vil det bidra med økt informasjon om utslipp i byer. Dette vil hjelpe myndigheter med å ta bedre og raskere beslutninger for å unngå spredning av farlige kjemikalier og/eller forurensning. For å lage slike nettverk må sensorene som benyttes være pålitelige, kostnadseffektive og robuste. En sensor som oppfyller disse kravene er den kjemiske kapasitive mikromaskinerte ultralyd transduceren (CMUT).

Denne oppgaven demonstrerer mulighetene for at en CMUT basert kjemisk sensor kan fungere som en gassdetekteringsenhet som kan klassifisere og kvantifisere kjemikalier i luft ved hjelp av maskinlæring. CMUT-en består av en rekke polymerbelagte celler som adsorberer ulike gasser. Adsorpsjonen forårsaker en frekvensforskyvning i sensorutgangen. Dette skiftet kan korreleres til identiteten og konsentrasjonen på det adsorberte kjemikalie ved hjelp av maskinlæring. For å benytte maskinlæring trenger man en referanseverdi på de ulike frekvensskiftene. Ved å ta referanseprøver, som er korrelert med sensoravlesningen, og som senere blir sendt til analyse i en gass kromatograf. Referanseprøveinnhenting viste seg å være en tidkrevende prosess, dermed ble en autosampler utviklet. Denne øker produktiviteten ved å tillate 30 timers kontinuerlig prøvetaking i tillegg til å redusere kostnader ved å erstatte menneskelig arbeidskraft. Autosampleren ble ikke brukt til datainnsamling i denne oppgaven.

I 4 uker ble CMUT-sensoren plassert i et drivhus og kontinuerlige prøver ble tatt. Målet med testen var at CMUTen skulle måle CO<sub>2</sub> konsentrasjonen i et plantebed for å teste: hvis sensoren kan måle CO<sub>2</sub> i lave konsentrasjonen i luft kan den og måle andre kjemikalier i luft. Relevante maskinlæringsmodeller ble trent og testet på den innhentede dataen. Resultatene viste at sensoren kunne måle CO<sub>2</sub> konsentrasjoner ned til 120ppm. Ulike maksinlæringsmodeller ble testet. Modellene skulle være i stand til å skille mellom høye og lave konsentrasjoner. *Det nevrale nettet med relu aktivering* hadde best resultat med en feil på ca 15 % for begge konsentrasjoner. Kvantifiseringen av dataen led av lite linearitet da datasettdrift ble observert under forsøket. Driften forskjøv frekvensen som virket ødeleggende på lineariteten. Kvantifiseringen var altså ikke pålitelig. En viktig erfaring er at maskinlærings modeller ikke blir bedre enn dataen modellen er basert på. Mer data må bli hentet inn. Videre forskning burde konsentrere seg om polymerene for å fjerne drift i dataen, hente mer data, minimalisere elektronikken samt å implementere en skydatabase for lagring og innhenting av data. Nevnte steg er indentifisert som viktige faktorer for å få til ett større sensornettverk med denne typen sensor.





## TABLE OF CONTENTS

Abbreviations .....	IX
List of tables .....	XI
Introduction .....	1
Overall research goals .....	2
1 Theory and basic concepts .....	3
1.1 Chemical sensing .....	3
1.2 Mass acoustic sensors .....	5
1.3 Technical review of existing chemical sensors .....	13
1.4 Machine learning with chemical sensors .....	17
1.5 Summary of theory .....	27
2 Chemical sensing experiment .....	29
2.1 Data collection tool .....	30
2.2 Data collection .....	37
2.3 Prediction .....	46
2.4 Results .....	47
3 Discussion .....	58
3.1 A model is only as good as the data its trained on .....	58
3.2 Assessing sensor behaviour during chemical sensing experiment .....	59
3.3 Comparing the CMUT sensor to end-user requirements .....	60
3.4 Comparison with available chemical sensor solutions .....	61
3.5 Autosampler allowing efficient sampling of data .....	61
4 Conclusion .....	64
5 Suggested Future work .....	66
6 Bibliography .....	69
Appendix .....	1
Appendix A: Linear regression models on a weekly basis .....	1
Appendix B: Code flow charts .....	3
Appendix C: Source code and data .....	6
Appendix D: Operation manual .....	23
Appendix E: Greenhouse test protocol .....	27



## ABBREVIATIONS

CMUT – Capacitive micromachined ultrasound transducer

GC - Gas chromatograph

SAW – Surface acoustic wave

QCM – Quartz crystal microbalance

MHz – Mega hertz

Ghz – Giga hertz

FBAR – Film bulk acoustic resonator

DMMP – Dimethyl methylphosphonate

MOF – Metal-organic framework

SVM – Support vector machine

RBF – Radial basis function

RMSE – Root mean squared error

TP – True positive

FN – False negative

DARPA – Defence advanced research projects agency

PEO – Polyethylene oxide

P4VP – Poly 4-vinylphenol

OV25 – OhioValley#25

PMMA – Polymethyl methacrylate

PVA – Polyvinyl alcohol

REF - Reference

NOK – Norwegian kroner

GUI – Graphical user interface

NMBU – Norwegian university of life sciences

FTIR – Fourier-transform infrared spectroscopy



## LIST OF TABLES

TABLE 1: COMPARISON OF THE DIFFERENT MASS-LOADING MEMS SENSORS.....	12
TABLE 2: POLYMERS USED AS COATINGS FOR THE CMUT SENSORS DURING THE EXPERIMENT AND THEIR SENSITIVITIES .....	39
TABLE 3: FEATURES AND REFERENCE USED AS INPUT TO THE MACHINE LEARNING MODELS .....	42
TABLE 4: DATA PREPARATION STEPS; DATA HOLD BACK, STANDARDIZATION AND ROW REMOVAL.....	43
TABLE 5: INFORMATION SUMMARY OF C AND $\gamma$ PARAMETER .....	44
TABLE 6: SUMMARIES OF MACHINE LEARNING MODELS FOR CLASSIFICATIONS USED AND FINAL SETTING FOR THEIR PARAMETERS.....	45
TABLE 7: ACCURACY AND CROSS VALIDATION SCORES OF THE DIFFERENT MACHINE LEARNING ALGORITHMS, LOGISTIC REGRESSION, LINEAR SVM, GAUSSIAN SVM AND NEURAL NETWORKS.....	48
TABLE 8: REGRESSION MODELS AND THEIR SCORES ENTIRE DATA SET.....	53
TABLE 9: LINEAR, LASSO AND RIDGE PREDICTIONS ON LEFT OUT DATA .....	55



## INTRODUCTION

Changes in the composition of the air we breathe can be harmful and in some cases lethal. As an example the world health organization predicts that 6.5 million deaths globally are caused by air pollution [4], another example is the chemical attacks in Syria causing 43 deaths related to exposure of highly toxic chemicals in April 2018 [5]. To help authorities and governments respond faster and make better decisions for avoidance of emissions and/or dangerous gases, real-time data need to be gathered. This data can be gathered by a network of chemical sensors.

A chemical sensor, often referred to as an electronic nose, consists of an array of sensors used to detect various compounds. Although chemical sensors have been researched for several years (the first electronic nose was developed in 1964 [6]) no commercial low cost sensor is available. Much of the difficulty in providing such a sensor to the market is making an inexpensive, precise and easily manufactured sensor that does not require pre-calibration. Current conventional sensor networks for detecting compounds are insufficient due to their limited number of nodes, large size and dependency on gas chromatographs and spectrometers [7]. To provide sufficient data, an increase in density of sensors is needed, thus contributing to the drive to develop new multifunctional chemical sensors.

The chemical sensor used in this thesis, is a capacitive micro machined ultrasound transducer (CMUT) for chemical sensing, developed by the Khuri-Yakub Ultrasonic group at Stanford University [8]. The sensor is inexpensive and has shown promising results as an electronic nose. Stedman, researcher in the Khuri-Yakub group, proved the sensor's ability to distinguish acetone, ethanol, methanol and water with 96% accuracy in a controlled environment [9]. Testing the sensor in a real-world situation exposing it to more complex gas mixtures like ambient air has still to be performed to evaluating its ability of environmental monitoring. This will be the main-focus of this thesis. Ambient air consists of 0.033vol% CO<sub>2</sub>[10], if the sensor is able to detect varying CO<sub>2</sub> concentration in air it can also detect other compounds at low concentrations.

To network of sensors that is able to detect compounds at low concentrations, a database of chemical concentration fingerprints is required, to train machine learning models. To create such a database, large sets of data are needed. Data for the library is gathered sampling sensor data and reference data providing the blueprint of chemicals present. A data collection tool is suggested in this thesis for efficient collection of reference data.

This thesis is written in collaboration with Stanford University and Fluent. A proof of concept of a CMUT based sensor as a gas detecting unit that can classify and quantify chemicals with machine learning in a real-world application is investigated.

## Overall research goals

**Overall goal:** Demonstrating a proof of concept of CMUT based sensor technology as a gas detecting unit that can classify and quantify chemicals with machine learning in a real-world application.

### Objectives:

- Design and build a mobile autosampler tool, for obtaining data points from a field test
- Compare the CMUT-sensors towards end-user requirements set by the Defense Advanced Research Projects (DARPA).
- Compare the CMUT sensor to available chemical sensor solutions
- Proof of concept testing of the CMUT sensor
  - Evaluate the viability of the test setup
  - Evaluate the sensor over a period of 4 weeks in a greenhouse
  - Evaluate its response by measuring the absolute CO<sub>2</sub> concentration in soil
    - If the sensor can detect low concentrations of CO<sub>2</sub> in ambient air, it can also detect other compounds
- Use machine learning to interpret data
  - Identify a suitable machine learning approach and model
  - Evaluate performance of machine learning models
- Suggest future work

### Limitations

- Only measure CO<sub>2</sub> gas in one plant bed within a greenhouse
- 4 weeks of intensive sampling
- Use of CMUT sensor only
- The autosampler is not used for sampling data in this thesis

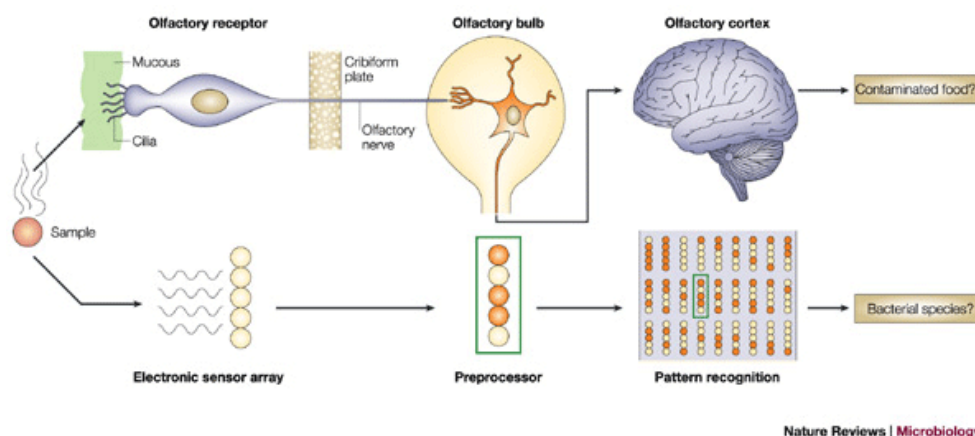


# 1 THEORY AND BASIC CONCEPTS

The theoretical base for this thesis will consist of three parts: sensor technology, commercial available products for chemical sensing and a machine learning part. Sensor technology focuses on technology related to the detection of compounds in air. After a brief introduction to chemical sensing and mass acoustic sensors an overview of mass acoustic sensor technology will be provided to help justify the sensor of choice – the CMUT. The technical review presents an overview of available chemical sensors to serve as contrasts to the CMUT sensor, and help identify important stepping stones for further development of the sensor. The machine learning chapter introduces relevant machine learning models and how they are applied for chemical identification. Sources of error in machine learning and ways to avoid them are also presented.

## 1.1 Chemical sensing

Human olfactory systems makes it easy to distinguish different smells, it takes seconds to distinguish a freshly baked bread from a rotten fruit. Recreating the sense of smell on a chip have proved to be challenging. The most common way to distinguish different compounds/smells is through gas and mass chromatography. Being selective and sensitive makes chromatography a popular method for compound identification. The downside is that these machines are large, and analysing tests are time consuming and expensive. To monitor larger areas like cities, farms or indoors the sensors must be inexpensive, robust and portable [11]. As a result, large efforts to find the best chemical sensor for this purpose it put forth.



**Figure 1: The human olfactory system compared with a chemical sensing approach, obtained from [12]**

Chemical sensors recognize different compounds due to their functionalization layers. The layers attract specific compounds. When adsorbed, the weight of the compound causes a shift. The shift is used as an input in pattern recognition algorithms, and hopefully the algorithm finds the origin of the compound (Figure 1). This process is similar to the human olfactory system, where olfactory receptors in the nose are used to detect compounds, when detected a signal is sent through a neuron to the brain for a recognition task. Chemical sensors have gained



increased attention due to their many usage areas such as; environmental protection and monitoring, bio sensors for recognizing chemical warfare agents, toxic gas recognition and spoiled food indicators [13]. No large commercial success has been seen, although the technology is in place.

In order to have a successful pattern recognition algorithm a large amount of data must be stored for the algorithm to train on. Recent developments in computer storage making it inexpensive and efficient to store large data sets, micromachining allows for great complexity on a tiny chip, helping to reduce the size of chemical sensors. And the computational power have increased over the past decade making it efficient to train machine learning models on large data sets. The technology is in place to make a chemical sensor that is inexpensive, robust and portable.

## 1.2 Mass acoustic sensors

Most chemical sensors are of type mass acoustic. Acoustic sensors consist of at least one vibrating element that creates acoustic waves. The acoustic waves are either propagated along the surface making it surface acoustic or within the material making it bulk acoustic. Mass loading of compounds will disturb the wave propagation which results in a frequency shift. The frequency shift can be used to identify the chemical [14]. By using chemical coating, often made up by polymers, selectivity is obtained. Polymer is a term for simple compounds joined together in larger chains. The chain is made up by smaller groups of atoms called monomers. In Figure 2 the formation of the polymer nylon from two monomers is illustrated. Monomers can range from 1-3 atoms to complex ring structures consisting of more than a dozen of atoms. Monomers link together and form the polymers [15]. For chemical sensing purposes polymers are functionalized to attract and release different molecules through adsorption, by introducing specific chemical groups [16].

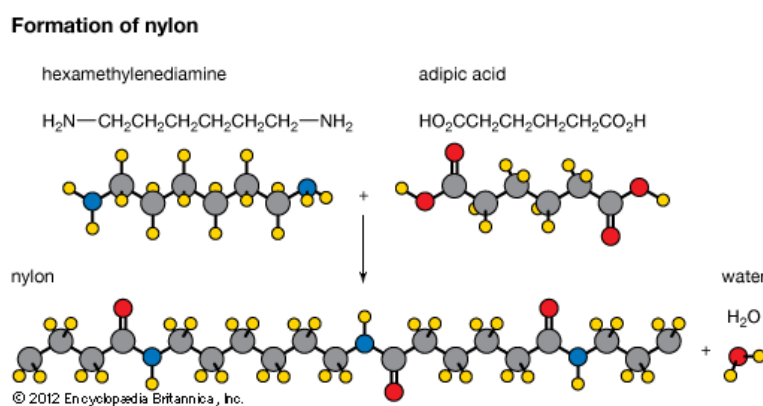
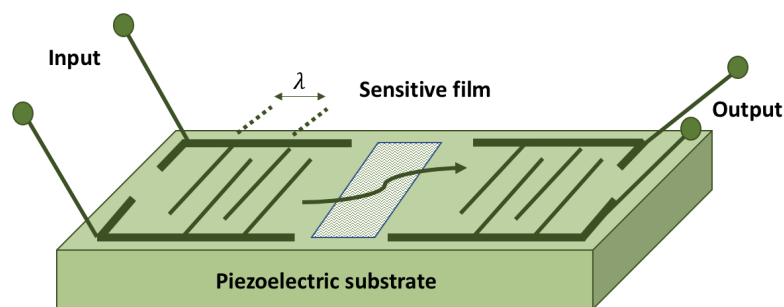


Figure 2: The formation of a nylon polymer from two monomers. Obtained from [16]

In the upcoming section, characteristics of five different acoustic sensors are describes; surface acoustic wave, quartz crystal microbalance, thin film bulk acoustic, resonant cantilever and a chemical CMUT sensor. The sensors are later compared to each other based on their sensitivity, detection range and cost.

## 1.2.1 Surface Acoustic Wave Chemical Sensor

Surface acoustic wave (SAW) chemical sensors exploit the concepts of surface acoustic waves produced by a vibrating element. Utilizing materials with piezoelectric properties, like Quartz, surface waves appear when the material is subjected to an electric field. The electric field is produced by InterDigital transducers (IDT). IDT converts an electric signal into a Rayleigh wave. IDTs operate both as transducers and as receivers [17]. Seen in Figure 3, the inputted electric signal is converted to a wave that propagates through the sensitive film. The wave travels to the IDT on the opposite side which translates the identity of the wave back to an electric signal. Interacting with the surroundings the wave velocity and amplitude will change [18]. Changes can be caused by mass loadings, mechanical stiffness or changes in the viscosity of the material.



**Figure 3: Simple design of a surface acoustic wave sensor modified from [19]. The forks represent the IDTs, a typical input is an AC signal and the sensitive film can be a polymer, the output is sent to a processor.  $\lambda$  represents the configuration width of the IDT, effecting frequency.**

The SAW sensors properties are so good that they are expected to meet the increasing demand of high performance chemical sensors in industries, military, pollution and emissions. Due to the sensors ability to sense both fluids and gases in addition to having an ultra-high sensitivity, detecting masses down to picograms [14]. It is also proven to be reliable, with an excellent sensitivity, selectivity and response time [20]. Thus capable of sensing numerous gases, chemical vapours and warfare agents. The SAW sensor has even been proved to work satisfactory even in harsh conditions [21]. Despite its many advantages the SAW sensor suffers from poor signal to noise ratio due to their high frequencies, as well as their complex and expensive fabrication [22].

## 1.2.2 Quartz Crystal Microbalances

Quartz crystal microbalances (QCM) uses the piezoelectric properties of the quartz crystal to sense mass changes to its surface. By applying an alternating current to the top and bottom electrodes the quartz crystal will vibrate. The vibration results in a wave that travels through the bulk of the sensors, with a resonant frequency operating in the megahertz (MHz) range [23].

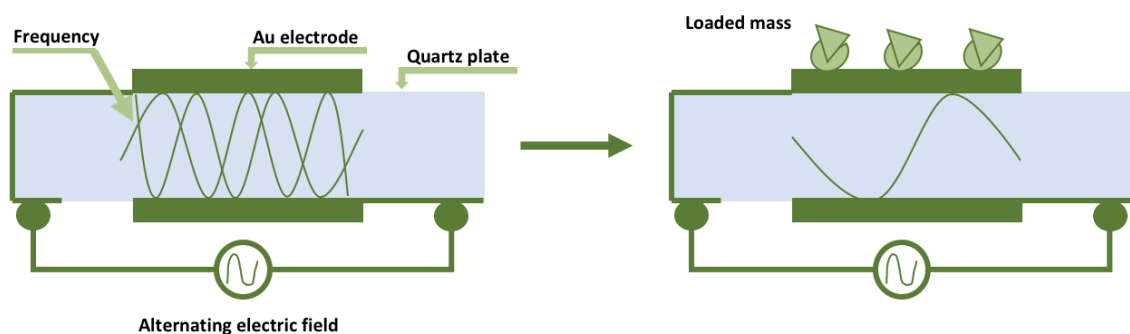


Figure 4: Working principle of a QCM. Showing how the resonant frequency changes when mass is added. Modified from [24].

The QCM is usually functionalized with recognition sites that attracts specific compounds. When a compound is attracted a disturbance in the resonant frequency can be measured. As seen in Figure 4, the bulk wave frequency is changed when a mass is loaded on to the sensor. The frequency change is proportional to the mass of the adsorbed material [25]. Molecular structural changes caused by chemical reactions on the surface can also be measured due to the many properties of the functionalization layer. Sensing in both liquid and gaseous environments makes it a versatile sensor [26]. Sensing mass changes down to 0.1 nanograms makes the QCM able to act as a tuneable chemical sensor for environmental and specific drug compounds [27]. Despite its high sensitivity and selectivity, the sensor has a complex fabrication procedure involving rare chemicals. Due to surface interference the QCM has a poor signal to noise ratio due [23].

### 1.2.3 Thin Film Bulk Acoustic Resonators

Thin film bulk acoustic sensors (FBAR) is a configuration of the bulk acoustic wave resonator (BAR) that exploits acoustic waves to detect gas-induced mass changes [28]. FBARs consist of a piezoelectric thin film that is sandwiched between two electrodes (Figure 5). By applying an alternating current to the electrodes a standing wave is created. The wave propagates through the material, reflected off the interfaces that have a large difference in their acoustic impedances [28]. The frequency ranges from 100 MHz to 10 GHz and is decided by the natural frequency of the piezoelectric material and its thickness [29]. When a chemical is adsorbed the stiffness of the sensing layer changes, causing a change in the resonant frequency of the piezoelectric layer. The frequency shift is measured used as an input to the machine learning model.

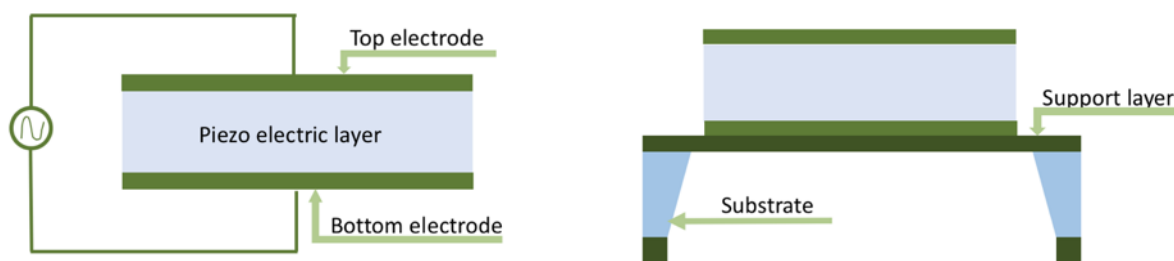


Figure 5: a) Schematic representation of BAR operation b) Schematic representation of a FBAR operation, thickness is not shown to scale. Figure modified from [29].

SAW and FBAR have quite similar working principles using waves to detect changes although it is important to note the difference between the waves. SAW sensors convert mechanical energy to Rayleigh waves. Rayleigh waves propagate along the surface of a material. While FBAR sensors convert the energy into standing waves, waves that are propagated in the entire material. Utilizing standing waves makes the FBAR advantageous to the SAW due to its insensitivity to surface contamination and adsorbates. It also has a better power handling. The main disadvantage of the FBAR is that it is dependent on large acoustic impedance mismatches on both sides of the reflector. Merging 3 or 4 acoustic dissimilar materials on top of each other [30] will create such a mismatch. Although creating the mismatch the process is complex to manufacture.

The FBAR sensor is no bigger than 500  $\mu\text{m}$  allowing multiple FBAR in a single array. An array of FBARs can increase the sensitivity and precision of the predicted compound [31]. Integration with other electronic devices is also possible due to its small size and simple operating procedure. These properties and the high frequency range making the FBAR more sensitive than the QCM and the SAW [31].

## 1.2.4 Resonant Cantilever Sensors

Resonant cantilever sensors are microcantilever devices (tiny diving boards) that detect changes in cantilever bending or vibrational frequency [32]. Cantilevers have two operation modes; static and dynamic (Figure 6). The static mode measures deflection caused by changes in temperature, surface stress or changes in electrical or magnetic fields. The Dynamic mode measures changes to the frequency of the cantilever, changes are caused by mass loading changing the damping of the cantilever [33]. When operated as a chemical sensor the top is often coated with a polymer, adsorbing specific compounds and causing different shifts. Either a deflection or change in frequency dependent on the operation mode.

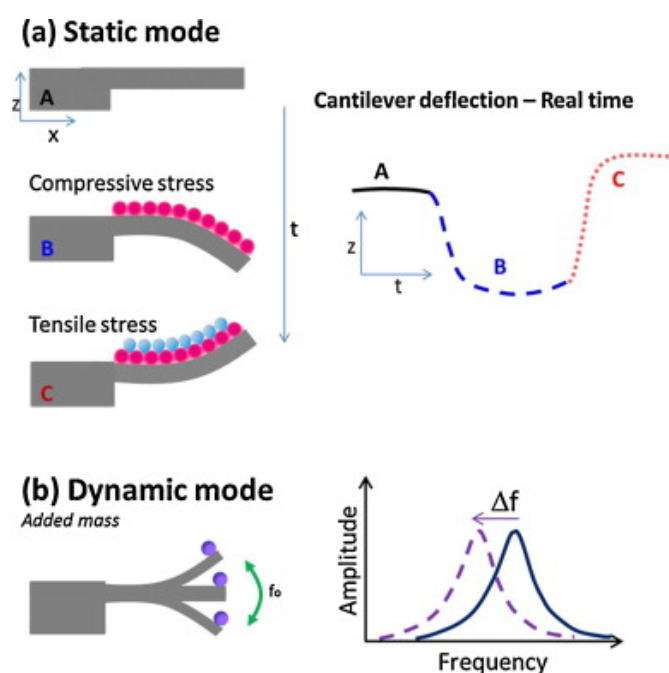


Figure 6: Different resonant cantilever operation modes. a) Static mode, cantilever deflection is measured in real time. b) Dynamic mode, frequency shifts are measured. Obtained from [34].

Cantilevers are fabricated by micromachining and can be made of materials such as silicon nitride, polymers and nanowires. The cantilever offers many advantages such as high specificity, sensitivity, simplistic and low cost fabrication [32]. Challenges the cantilever sensors face are sensitivity to vibrations and high acoustic fields, measurement drift, and change in the response of the sensor [33]. Although facing many challenges arrays of resonant cantilevers has been proven to work as an chemical sensor, distinguishing certain natural flavours and alcohol mixtures [35].

## 1.2.5 Chemical CMUT Sensor

Capacitive micro-machined ultrasonic transducers (CMUT) can be thought of as structures that send and receive acoustic signals in the ultrasonic range. Due to their large bandwidth, easy fabrication and the large potential of integration with electronic circuits, they are widely used for medical imaging and chemical sensing purposes [8].

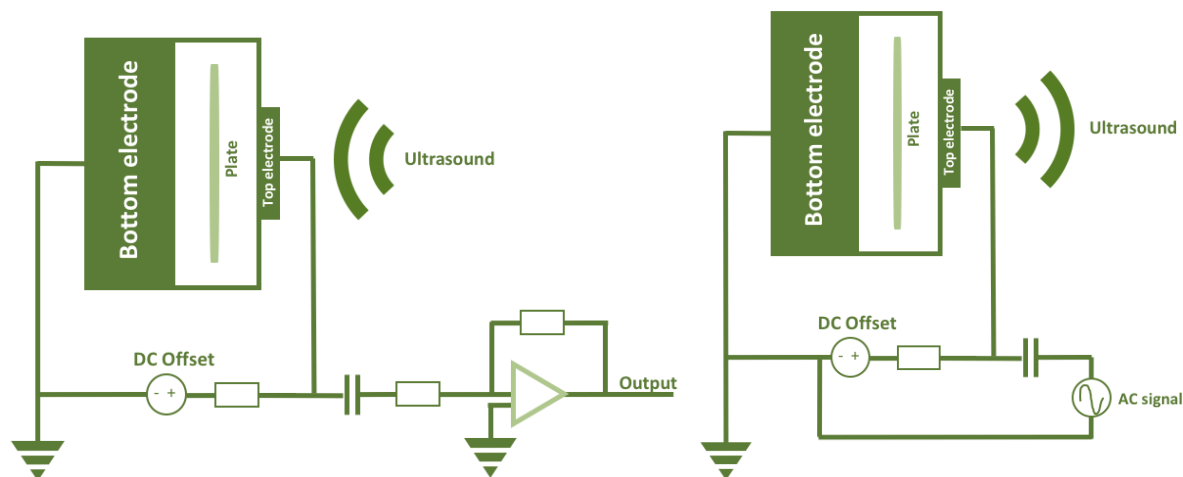


Figure 7: General CMUT operation principle a) ultrasound receiver b) ultrasound transmitter. Modified from [36].

The main components of a CMUT are two electrodes, bottom and top, separated by an insulating layer and vacuum sealed gap. The components are attached in parallel with a capacitor cell, seen in Figure 7. By generating an electronic potential between the two electrodes ultrasound is emitted due to the electrostatic force between the fixed and movable electrode [37]. When the CMUT acts as a receiver the procedure is opposite. This trait allows the CMUT to operate in a wide range of resonance frequencies; frequencies where the response amplitude is a relative maximum [38]. The large frequency range results in a large bandwidth of operation making it able to detect mass changes down to attograms ( $10^{-18}$ ) on its surface [39].

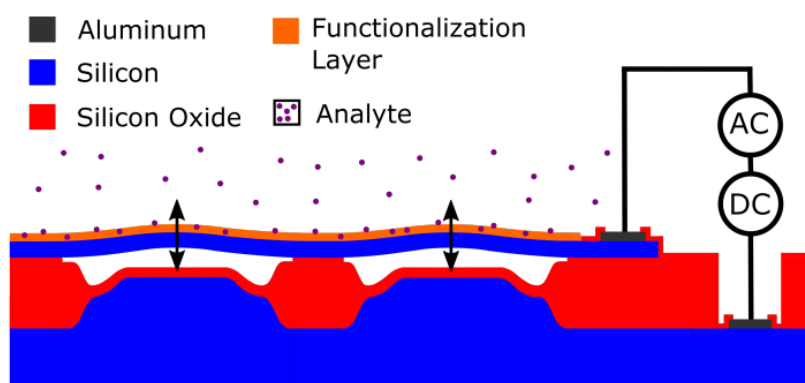


Figure 8: Schematic of a CMUT chemical sensor, the red functionalization layer is silicon oxide which attracts a certain compound. Obtained from [9].

A chemical CMUT is a CMUT with a functionalization layer that interacts with compounds in the surrounding environment. The functionalization layer often consists of polymers that





absorbs specific compounds. Making the sensor selective to certain compounds. The small size of the CMUT makes it possible to arrange multiple sensors in large arrays. Different sensors have different coatings which enables multiple compound detection [40].

Selective sensing, easy fabrication and reliable operation in harsh environments makes the chemical CMUT sensor a strong contender for a portable, inexpensive and robust sensing apparatus. Although drift in the data over time due to temperature and pressure differences has been observed [9]. This problem was handled in [41], where temperature and pressure sensors were implemented on the array board for correction of the data.

## 1.2.6 Summary of mass acoustic sensors

For enabling scalability, chemical sensors should be robust, portable and be inexpensive. Robust is in this case the sensor sensitivity and that it is able to detect small changes in large areas. These traits are compared for the sensors presented in the theory chapter. A score for each sensor is put forth. The scores for range and cost are educational guesses based on personal communication with Quinten Stedman, PhD candidate in applied physics at the Khuri-Yakub group. Sensitivity scores are based on papers that test the sensors response to the recognition compound of mustard gas, dimethyl methylphosphonate (DMMP). Although not comparable to all gases it gives an overall indication of how well the sensitivity of the sensors for a chemical warfare agent. The scores are presented in Table 1, ranging from 0-10 where 10 describes excellent qualities and 0 represents poor qualities. The Range of detection is the volumetric area the sensor can sense different compounds in, and will also depend on the chemical being tested. Cost scores are based on the processing and the machining process for the different sensors.

**Table 1: Comparison of the different mass-loading MEMS sensors.**

Structure	Sensitivity	Range	Cost
SAW	7 [3]	3	7
QCM	4 [42]	5	4
FBAR	6 [43]	6	9
Resonant cantilever	8 [44]	5	9
Chemical CMUT	7 [45]	9	9

## 1.3 Technical review of existing chemical sensors

A technical review of existing chemical sensors is put forth, to contrast the chemical CMUT sensor and to identify important steps for further commercialization.

### 1.3.1 FBAR Sensors – University of Cambridge

In 2012 researchers at Cambridge developed a Film Bulk Acoustic Resonator (FBAR) device that measures temperature and mass-loadings simultaneously. The sensor can operate both in air and liquid environments where the standard functionalisation chemistry allows for FBAR sensing [46]. The parallel sensing is a result of the careful design of the resonator structure that allows for deconvolution. By having such a control over the microstructure, a resonance that is not affected by liquid on its surface is allowed for.

### 1.3.2 QCM Sensor – Matrix sensor

Matrix sensors use a metal-organic framework (MOFs) to adsorb chemicals and a quartz resonant mass transducer for measurements. MOFs are recognized by their open framework (Figure 9 a) and can be compared to a porous material, almost like a molecular sponge. MOFs are used for gas storage, purification, separation as well as a catalyst [47]. MOFs are also easy configurable which allows for specific sensing. Specific sensing, low power consumption and small size characterizes the matrix sensor. Matrix sensors inc specializes in detecting carbon dioxide, methane and toxic gases [48].



Figure 9: a) A metal-organic framework from [49] b) the matrix sensor from [48]

Inside the black box (Figure 9 b) there is an a pressure and temperature sensors as well as an array of matrix sensors. The pressure and temperature sensors are used for self-correction of drift due to environmental conditions like temperature and pressure changes. Matrix sensor inc claims that their sensor will cost less than 30\$, use less than one quarter watt and be smaller than 2,5x2,5 cm [50]. Matrix sensors inc have an exclusive license to the most fundamental technology, and they have two issued patents and four applications in process.

### 1.3.3 QCM sensor – “Open QCM” by NovaTech

The OPEN QCM sensor is based on quartz crystal microbalance technology which allows the sensor to measure mass loadings at a molecular scale, the sensor is seen in Figure 10 b. OpenQCM is applicable in the field of chemistry, biology and material sciences and it is promoted as an entire laboratory on the surface of a crystal [51]. The sensor can measure mass variations down to 0.1 nanograms in air, vacuum and liquid environments.

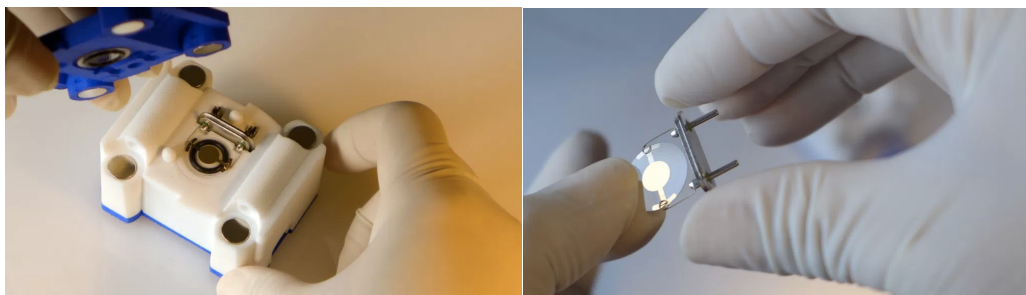


Figure 10: a) Open QCM sensor box b) QCM sensor chip both from [52]

Effective measurements of binding events happening on the QCMs surface makes it a powerful tool for sensing DNA hybridization and specific drug compounds. It can also be used as a tunable gas sensors monitoring air quality in ambient conditions. OpenQCM focus on being an open source platform to engage as many crowd scientists as possible. This will help with the improvement of the sensor as well as finding new appliance areas. Research papers using the OpenQCM sensor have been published in various fields ranging from detecting explosive vapors to detection of parathion in water (type of pesticide) [53][[54]. Although the many usage areas the data researchers obtain from the sensors contains a lot of noise. There has been little activity since 2017, but the sensor is still available in their online store. Starter kits are available at \$399, mounted and tested versions cost \$499, seen in Figure 10a) [55]. The high price is due to the expensive materials, gold and titanium, which is used in fabrication of the sensor chip.

### 1.3.4 Chemi-Resistors – CHEMISENSE

ChemiSense offers a chemical and particle sensor, providing solutions for the industry, commercial- and residential markets. In 2014 they started a crowdfunding campaign for a wearable bracelet that measures the air pollutants in a user's hyperlocal environment [56]. The data from the bracelet (Figure 11 a) is sent to the user's smartphone where machine learning is employed to analyse the data (Figure 11 c). If a dangerous threshold of air pollutants is measured the user is notified. ChemiSense also want to use the data to make a heat map of air quality so inhabitants of large cities can avoid heavily polluted areas or to help authorities respond better to pollution.

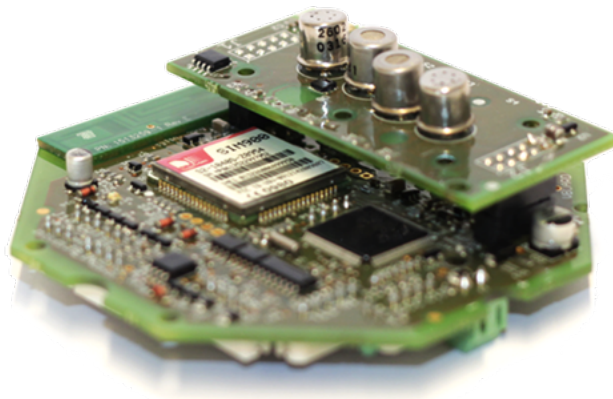


**Figure 11: a) Early prototype of the CHEMISENSE bracelet. b) App that provides real-time data processing both from [57]**

ChemiSense uses chemi-resistors to measure pollutants. Chemi-resistors consist of a polymer treated with charged nanoparticles of carbon. If a certain chemical is present the polymers swell, changing the resistance of the circuit [57]. ChemiSense plans to launch their indoor sensor, measuring 10 ppm, in 2018. Although the measurements are not as sensitive as first expected, they are aiming for 100 ppb resolution. The sensor can detect about a dozen chemicals like benzene, hexane, nitrogen dioxide and carbon monoxide [58]. The last recorded activity was in 2016 when their software engineer published his master thesis [59].

### 1.3.5 Port of Rotterdam – We-nose network

The port of Rotterdam has implemented a network of 250 e-noses, chemical sensors, for continuous monitoring of the surrounding air. Four semiconducting sensors make up the e-nose, developed by Common Invent, giving responsiveness to specific gases [60]. Patterns are created by the sensor when exposed to the reactive gases giving each gas a specific fingerprint. Through an internet connection the sensors upload the pattern data to a cloud database. In the cloud database, the uploaded data is compared to fingerprints of known chemicals. If a similarity is found an operator is noticed, either through a phone or desktop application [60]. The operator can further investigate the source of such a change. Specifications on the eNose is: Length = 11.9 cm, width 16.0 cm and it requires 2W for operations.



**Figure 12: Common invents e-Nose used in the sensor network in at the port of Rotterdam. The four metal cylinders on top are the semiconducting sensors image from [60].**

## 1.4 Machine learning with chemical sensors

Machine learning is defined by nVIDIA, a world leading AI provider, as;

*“Machine Learning at its most basic is the practice of using algorithms to parse data, learn from it, and then make a determination or prediction about something in the world.” [61]*

In this thesis machine learning is used to analyse sensor data, learn from the data and determine the chemicals present and the concentration of the chemical when the data was sampled. The two main tasks are: identify the chemical present (qualitative measurement) and determine its concentration (quantitative information).

For a machine learning algorithm to classify a chemical present, several samples of the sensor response to the chemical needs to be taken. After the sampling, features representing the chemical are extracted. Features are for example; the specific frequency shift of the sensor or a distinct response at a given time. Features are used as input to the machine learning models. Giving the model data to find patterns in that correlate with the reference data. The patterns found are used to define limits between high and low concentrations, or linearities amongst the data which can be used for quantification purposes.

In the specific case of this thesis, the collected data is the frequency shifts from the sensor and concentrations from the GC, at given sampling times. Machine learning models are employed to find a relationship between these quantities. The main-focus of the machine learning models is to minimize the model error and make the most accurate prediction. These models are heavily based on statistic tools, for example linear regression. As machine learning and statistical modelling lie very close, it is important to separate them from each other. Machine learning finds relationships through many iterations, whilst in statistical modelling the modeller needs to understand the relation between variables before using them in a statistical model [62]. Statistical modelling is also bound by linear boundaries. Machine learning captures patterns beyond the linearity and boundaries. Often allowing multiple boundaries in one prediction. This section presents machine learning models such as logistic regression, support vector machines and artificial neural networks performing classification tasks and regression models such as linear, lasso and ridge regression performing quantification tasks. First a brief introduction to data pre-processing is given.



### 1.4.1 Data pre-processing

Before using the data to train machine learning models, the data needs to be pre-processed. Strategies for pre-processing the samples vary amongst data scientists; some use the samples directly while others scale the data down. Directly utilizing the data results in a multidimensional classification and regression problem. If the corresponding dataset is large one might be prone to overfitting as well as using a lot of computational power to solve for the right algorithm. Sampling down the data results in a multivariate representation of the data, and the corresponding model does not take the temporal structure into account [63]. Normalization and standardization is also common when pre-processing the data, dependent on the distribution of data. Normalization is often applied when the data vary in parameters, for example if one feature is age and another one is income it is usual to normalize them into the range between 0 and 1. Standardization is applied when the features vary in size, if one response has a much higher value than the smaller response scaling the features to a unit variance between -1 and 1.

### 1.4.2 Supervised machine learning

Supervised machine learning is recognised by mapping an input to an output, and the fact that the data scientist acts as a teacher for the algorithm guiding it to its conclusions. One can think of supervised learning as taking a labelled set of data and extracting as much information as possible from the set [64]. Using the extracted information to predict an unlabelled input. In other words; function approximation as seen in eq [1]

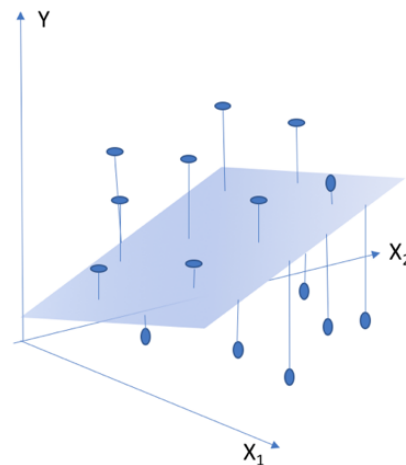
$$y = f(x) \quad [1]$$

Bzdok, Krzywinski and Altman (2018) defines supervised learning algorithms as “*algorithms extract general principles from observed examples guided by a specific prediction objective*”[65] . In this thesis the learning is supervised due to the use of input variables to predict a given concentration. The output or prediction objective is known.



### 1.4.3 Regression

Linear regression is both a machine learning and statistic tool as it describes the relationship between the input and output variables, based on the assumption that there is a linear relationship between them. When multiple features are used as input the problem is of a higher dimension, and the line describing the relationship is thus called a *hyperplane* [66]. A hyperplane can be thought of as an  $n-1$ -dimensional plane dividing  $n$ -dimensional space into two pieces.



**Figure 13: Hyperplane with ordinary least squares fitting in a 3-dimensional space. Circles represent data points and the lines represent distance to the plane.**

The ordinary *method of least squares* is used to compute coefficients for the multivariate regression. This is done by minimizing the squared distance from the points and the suggested hyperplane, seen in Figure 13. To optimize the linear regression *lasso* and *ridge regression* techniques are used. Lasso and ridge regression has as a goal to minimize the coefficients put forth by the least squares method. Implementing lasso and ridge helps avoid overfitting of models and improves prediction accuracy [67]. Lasso and ridge makes a regression model “optimized for prediction”.

*Ridge regression* uses the same formula as ordinary least squares regression to make its predictions. The only difference is the penalties added to the loss function. Ridge regressions adds a penalty equal to the squared magnitude of a coefficient, as well as satisfying an additional constraint. With the constraint being that the coefficients should predict well on the training data, and being as close to zero as possible [66]. Indicating that each feature has little effect on the output. This is done by adding a penalty equivalent to square of the magnitude of coefficients. Ridge regression makes a trade-off between a simplicity and performance of the model.

*Lasso regression* also restricts the coefficients to be close to zero, but uses a slightly different way of regularizing the coefficients. With lasso regression some of the coefficients are exactly zero [66], meaning that the model ignores some of the input features. By ignoring less important features the most important features are revealed. This is done by adding penalty

equivalent to the absolute value of magnitude. Lasso regression makes the model easier to understand by revealing the most important features [66].

#### 1.4.4 Logistic Regression

Logistic regression is a linear model used for classifying binary problems. It outputs the probability of an input belonging to any of the classes suggested by the user. The input is usually continuous while the output is discrete, describing the probability of a class membership. For example, predicting if a high grade 1, or a low grade 0 is given on a master thesis the learning function (eq [2]) obtained from [68], is used to predict the probability of the input, this thesis, belonging to the high class.

$$h_{\theta}(x) = P(y = 1 | x) = \frac{1}{(1 + e^{-\theta^T x})} = \sigma(\theta^T x) \quad [2]$$

where  $h_{\theta}(x)$  is a function parametrized by  $\theta$ ,  $x$  is the input vector and  $\theta^T x$  is fitted to the range (0,1) by the sigmoid function:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \quad [3]$$

so that  $h_{\theta}(x)$  can be interpreted as a probability with  $z$  being loss which iteratively updates the  $\theta$  function [68]. In words: logistic regression models the probability of an event instead of the target [69]. Advantages of the logistic regression is that it does not make any assumptions about distribution of classes in feature space. Another advantage is that it has a quick training time and it is resistant to overfitting. Its disadvantage is that it has a linear decision boundary.

### 1.4.5 Support Vector Machines

Support vector machines (SVM) uses the extremes of the dataset to set boundaries for the classification process. These boundaries are called support vectors and are utilized to find the optimal hyperplane for the dataset. The hyperplane is the linear separation line that classifies the two groups of data. SVM specialises in finding the optimal hyperplane of the dataset. Figure 14 a) shows that many hyperplanes can be set to segregate the data, but only one is optimal. An optimal plane is a plane that has a maximal distance from the support vectors (Figure 14 b)).

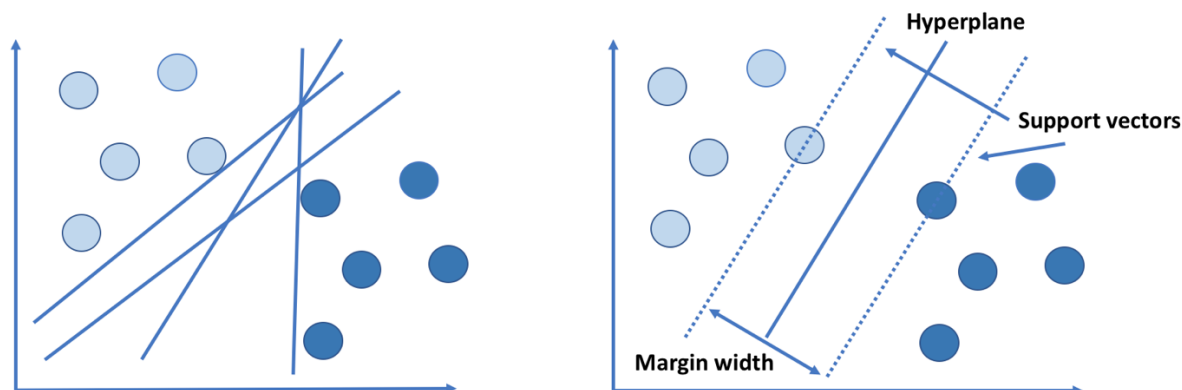


Figure 14: Classification task for two classes of data, light and dark blue spots a) Many hyperplanes can be defined between the two classes, illustrating how hard it can be to find the optimal plane. b) By using the margin method, the hyperplane with the largest margin is found

Chances of misclassification is little is if a maximal distance between the planes is set. As a result the model is robust, enabling it to respond well to data that it is not trained on [66]. To optimize the model the slack variable can be tuned to balance the bias and variance. Little slack allows for errors while fitting the data. This results in a larger margin which might allow the model to solve non-linear problems. Larger margins indicate that the classes are less similar.

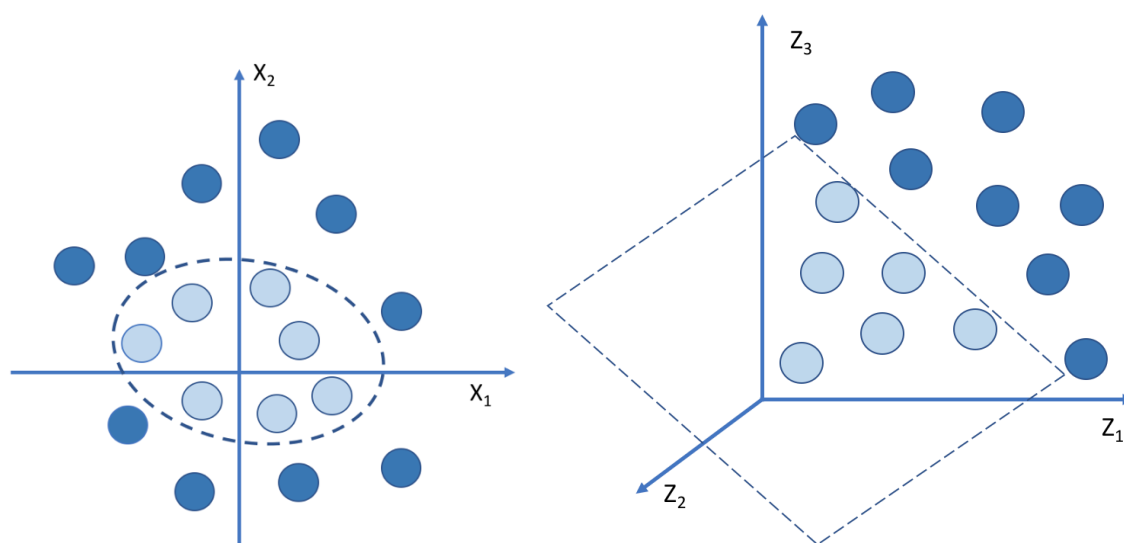
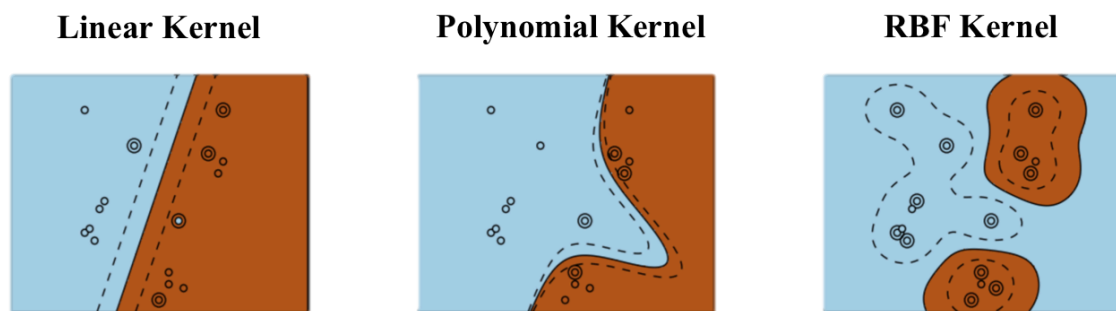


Figure 15: Transformation from input space to feature space by a kernel function of two different classes of data.

Usually the SVM is kernelized for non-linear problems, for example the data set in Figure 15 does not separate well by a single line. To solve this a kernel is used, kernels are mathematical functions that measures how similar two vectors are and project features into a higher dimensional space. Where the data can be linearly separated [66]. The kernels can be linear, polynomial or radial basis function (RBF). Figure 16 shows how the different kernels separate data, the polynomial and rbf kernel are better at separating non-linear data. SVM are iterative training functions aiming at minimizing an error function. SVM balances the trade-off between margin width and misclassification. Only a small subset of training points is needed, since the model uses few points to decide the support vectors, defining a classification rule [66].



**Figure 16: Presenting three different kernel types for a SVM model. The stippled lines present the support vectors while the line represents a hyperplane dots represents data. a) Shows a linear kernel b) shows a polynomial kernel and c) shows a Gaussian kernel. Image from [70].**

### 1.4.6 Artificial Neural Networks

An artificial neural network is a learning method developed from the idea of simulating the human brain and is a sophisticated way of pattern finding in data with single or multivariate analysis. Dr. Robert Hecht-Nielsen the inventor of the first neurocomputers defines a neural network as “*a computing system made up of a number of simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs.*” [71]. A neural network commonly consists of an input layer, a hidden layer and an output layer (Figure 17 a).

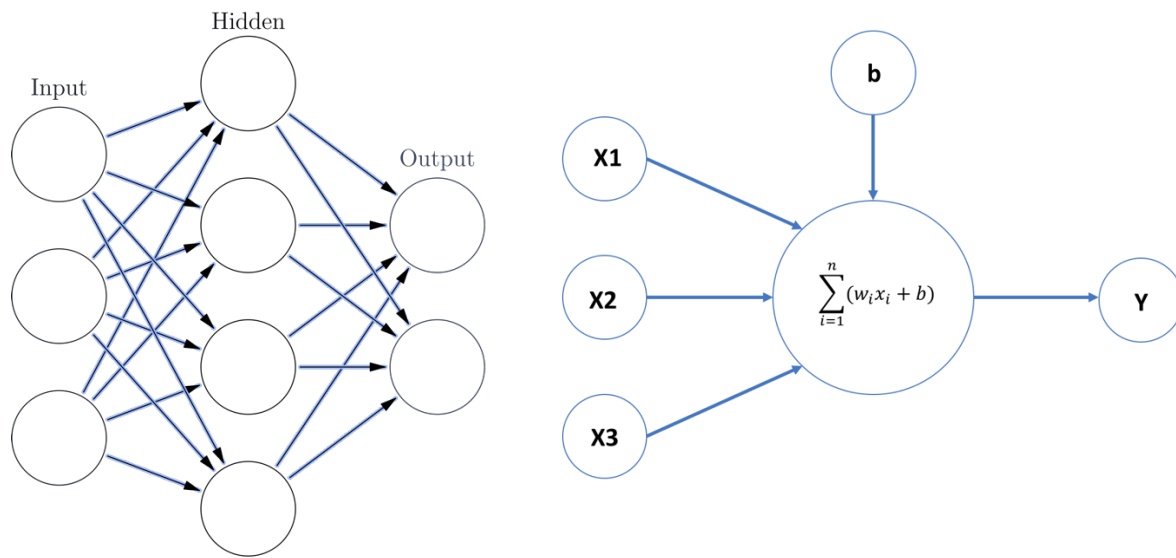


Figure 17: a) A neural network with an input, hidden and output layer consisting of multiple neurons. b) A simple neuron with inputs X1 to X3 with corresponding weights, a bias, b, and binary output Y.

The layers consists of multiple processing elements called neurons. Neurons take multiple inputs and outputs a binary number, Y, as presented in Figure 17 b). The process is widely used to make classification algorithms where the algorithms learns to classify based on given inputs from each classification category. A commonly used classification equation is presented below;

$$y = \begin{cases} 1, & \sum w_i x_i + b > 0 \\ 0, & \sum w_i x_i + b \leq 0 \end{cases} \quad [4]$$

$w$  is the weight,  $b$  is the bias and  $x$  is the input variable.

If the sum is above zero the perceptron outputs 1, if it is less than or equal to zero the perceptron outputs a zero. Weights,  $w$ , are added as the importance of each input value, while the bias,  $b$ , is a measure of how easy the perceptron outputs 1. A larger bias increases the chance of activating the perceptron. Weights and biases are learned parameters that must be tuned and updated to improve the output. Small changes to the parameters are recommended so one does not get stuck at local minimum or maxima, gradient descent is a commonly used for tuning the parameters [72]. Outputting binary values proves challenging as small changes in the parameters results in larger changes in the behaviour of the network. To solve this problem the

sigmoidal function is used in the final output layer so that a smoother output is achieved. Allowing gradual adjustment of the weights and biases. Sigmoidal functions refer to functions with S-shaped curves examples of these are softmax, tanh and the well-known sigmoidal function [72].

### 1.4.7 Evaluation metrics

Evaluating model performance is key to finding the best machine learning algorithm, evaluation metrics are used to evaluate different models. The metrics used, depend on the algorithm and are different for classification and regression models. For regression models, the mean squared error and the R-squared score is used which is an estimate of the distance between the line and points. For evaluating classifications models, the implementation of the model needs to be considered. For example, take a model that diagnoses patients as sick or healthy, it is worse to diagnose a sick patient as healthy (false negative) than it is to diagnose a healthy patient as sick (false positive). To evaluate how the models classify features confusion matrixes, accuracy, precision, recall, F1- and F-beta scores are used. Another important evaluation metric is the time the model uses to put forth a prediction, time is often a critical measure of model performance. A confusion matrix, accuracy and cross validation and is used to evaluate classification models. While the root mean squared error and the R-squared score is used to evaluate the regression models.

*The confusion matrix* presents the accuracy of a model with two or more classes. Returns a table that presents predictions on the x-axis and accuracy outcomes on the y-axis where the cells represent the predictions made [66]. Confusion matrix gives a visualization of the performance of a machine learning algorithm. The diagonal of the matrix represents correct predictions. As seen in Figure 18 a) the correctly tuned model gives perfect predictions, and the poorly tuned model in Figure 18 b) can correctly classify water but it is struggling to classify the other classes. The confusion matrix is a good tool for visually inspecting different machine learning algorithms and to inspect what classes are confused with one another.

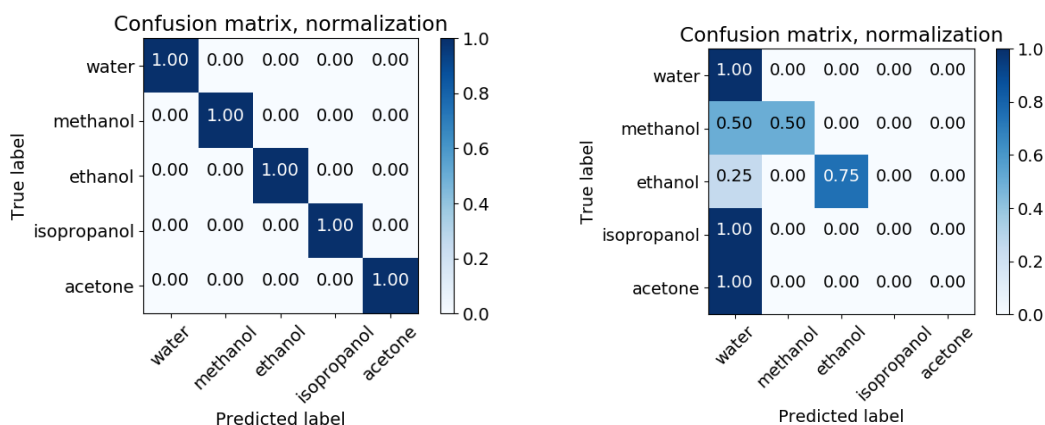


Figure 18: a) Confusion matrix with a well-tuned algorithm. b) Confusion matrix for bad-tuned algorithm



*Accuracy* is the ratio of the correctly classified points to the total number of points. Suitable if there is an equal number of observations in each class, if data is unbalanced accuracy gives a false interpretation [66]. Scoring from 0-1, where 1 is the best.

*Cross validation* test how the machine learning models generalizes to unseen data by holding out different subsets of testing data. Training the model on different subsets and validating the model on the remaining data. This is an iterative process, testing different subsets until there is no more left. The resulting score is an average of the accuracy of all the subsets [73].

The *root mean squared error (RMSE)* Sum of the absolute differences between predictions and actual values, in other words; how wrong the predictions are [74]. 0 represents a perfect prediction.

The *R-squared score* indicates what portion of the total improvement opportunity our model covers. Dividing the error of different models using their mean squared error as comparison ground [75]. 1 indicates that the model is perfectly fitted.

### 1.4.8 Sources of error in machine learning and ways to avoid them

In machine learning the overall goal of the algorithm is to find the best fit from the training data, if unsuccessful poor results are obtained. The most common cause of poor performance in machine learning comes from overfitting the model. Overfitting the model is a result of the model learning on all the noise and detail in the dataset instead of generalizing the data points. As seen in Figure 19 c) the proposed model will not respond well to new data. To check for overfitting different methods of accuracy test are developed. For example, cross validation which uses different subsets of the data for testing the model. Cross validation will give an insight on how the model behaves to unseen data, by holding one data point out and testing all the others. Repeating for all test points available until a generalized error estimate is obtained [76]. Another good practice for avoiding overfitting is keeping a couple of samples for the dataset and test the samples on the algorithm at the very end of the project, this will give an objective idea on the model performance. Another common mistake is underfitting the model, seen in Figure 19 a).

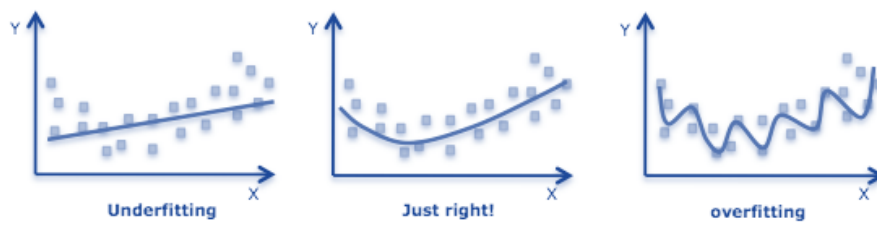


Figure 19 Different types of models fitted to data, a) is under fit b) is just right and c) is overfitted

Underfitting the model means that the model does not generalize well to the training or testing data. It is easy to detect when it is present the algorithm chosen cannot describe the underlying phenomena in the data seen. If no good predictions are put forth by the model it is most likely underfit, failing to describe the underlying phenomena [66].

### 1.4.9 Importance of good data

Machine learning models are only as good as the data they are based on. Bad predictions are often a result of bad data. With bad meaning that the collected dataset does not represent the true nature of the problem. Datasets used in machine learning should preferably be evenly distributed among all concentrations and contain a large amount of data for better generalization of machine learning models.





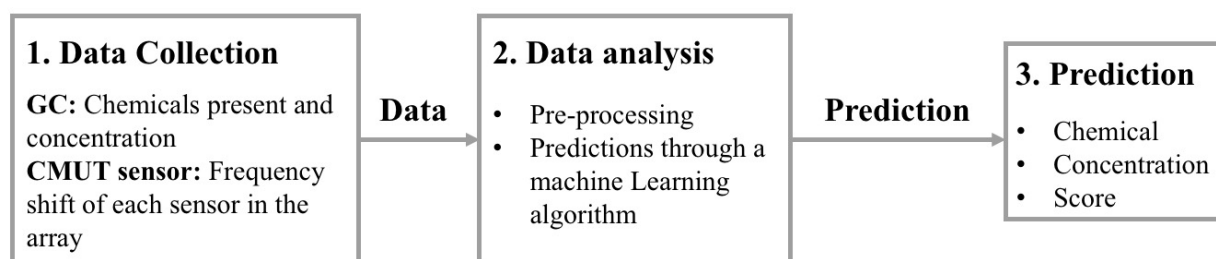
## 1.5 Summary of theory

Relevant sensor technologies are presented in the theory chapter as well as commercial available products, a theoretical background for machine learning models is put forth for identification of relevant models. After presenting the different sensors technologies they are compared. Justifying the sensor chosen – the CMUT. The review on different sensor technologies are later used for comparison purposes in the discussion. Relevant machine learning models presented are: logistic regression, support vector machines and artificial neural networks for classification purposes, lasso and ridge regression models are presented for quantification purposes. Presented models are compared when used to classify and quantify the data.



## 2 CHEMICAL SENSING EXPERIMENT

In this section, the three main steps of the chemical sensing experiment in the greenhouse are put forth, presented in Figure 20. The first step is data collection. Data is collected for the gas chromatograph (GC), and used as a reference for the frequency data produced by the CMUT sensor. The second step is data analysis. This part consists pre-processing of the data in addition to several machine learning models which provide a prediction from the collected frequency data, classifying it as high or low and determining its concentration. The GC samples are used for training the algorithms. The third step is the prediction. The machine learning model comes with a prediction and a score that reflects the accuracy of the prediction. For predicting concentration, it is how far the measured points are from the suggested hyperplane. For classification, it shows how well the model guesses the different classes.

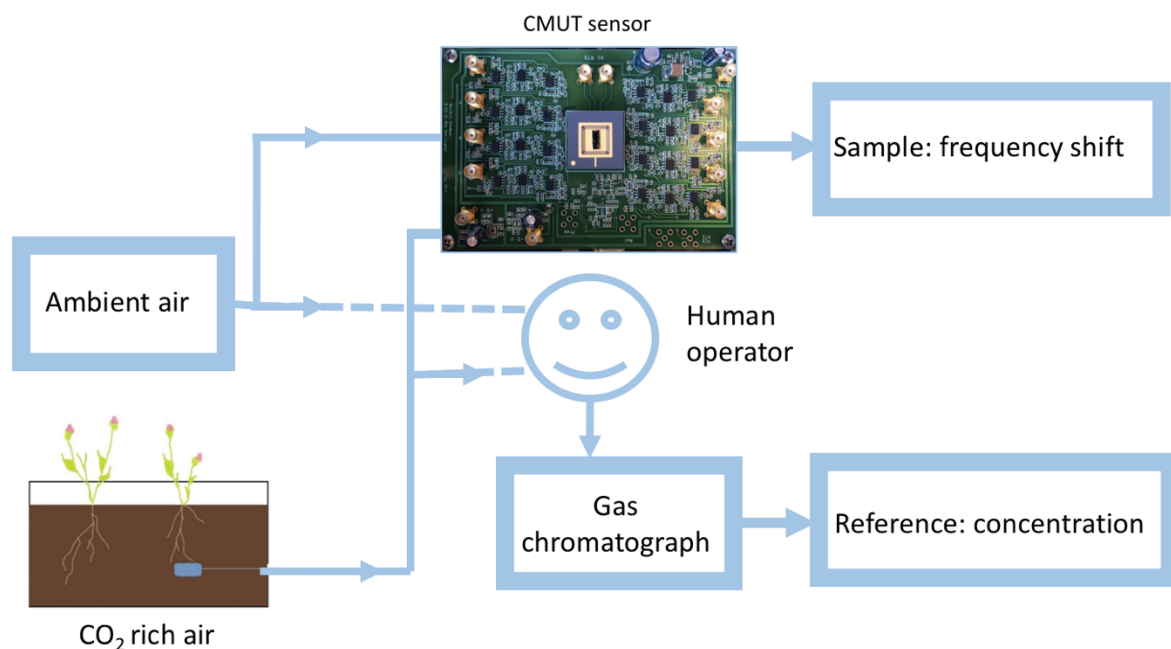


**Figure 20: Information flow for the chemical sensing experiment, first sensor and air sample date are collected, further the data is analysed with machine learning before a classification and concentration prediction is proposed with a score which reflecting its accuracy.**

The experiment is designed as a proof of concept, testing the following idea: *If the sensor can detect CO<sub>2</sub> at low concentrations in ambient air, the sensor can also detect other compounds in ambient air.* Following factors are identified as success factors: functional layers selected do have a useful response to CO<sub>2</sub> and reliable operation of the CMUT over a period of 4 weeks. These will contribute to the further learning of the CMUT sensors behaviour over time. As an evaluation of performance, the CMUT sensor is compared to end-user requirements defined by the Defense Advanced Research Project Agency (DARPA). As a US military supplier, DARPA's standards reflect the precision and accuracy demanded, hence reflects the customer's standard for chemical detection systems. The standards set by DARPA for a chemical detection system are; the sensor should be able to recognize at least 5 different chemical warfare agents, determine each chemical with a probability of 85% and the probability for a false alarm should be 10<sup>-5</sup> % [77].

## 2.1 Data collection tool

As a tool for data collection an autosampler was developed, allowing efficient and 24/7 sampling without the need of human operators. Providing continuous samples for analysis which is processed by machine learning models. As seen in Figure 21 the human operator can be replaced by the autosampler, for inexpensive and more efficient sampling. By replacing the human operator one increases the quality of the sampled data as well as obtaining more samples. An example of increasing the quality is that the autosampler works on a timer and a predefined path, avoiding mixing the vials and uneven sampling periods. Human operators are more prone to mixing mistakes and uneven sampling periods. The autosampler was not used during the data collection for the thesis as the samples were collected during winter 2017. As a proof of concept test the autosampler was stress tested, sampling the same vials 15 times in 1 hour.



**Figure 21: Data collection: samples from the earth and air are first flushed through the sensor to obtain the frequency shifts, before a human operator fills glass vials with air samples from the air and earth and sends them to a gas chromatograph to obtain reference concentrations.**

### 2.1.1 OpenTron Autosampler

For machine learning large datasets are an advantage due to the possibility of thoroughly testing and tuning the algorithm. Previously, data has been collected by hand, a time consuming and expensive effort. Each sample takes 10 minutes, and the cost for manual operation is 210 NOK per hour [78]. To sample 1 000 points, which is a large dataset, will cost 350 000 NOK. A suggested replacement is the OpenTron platform, which cost about 40 000 NOK [79]. OpenTron is an open source pipetting robot specialised for automating pipetting experiments (Figure 22).

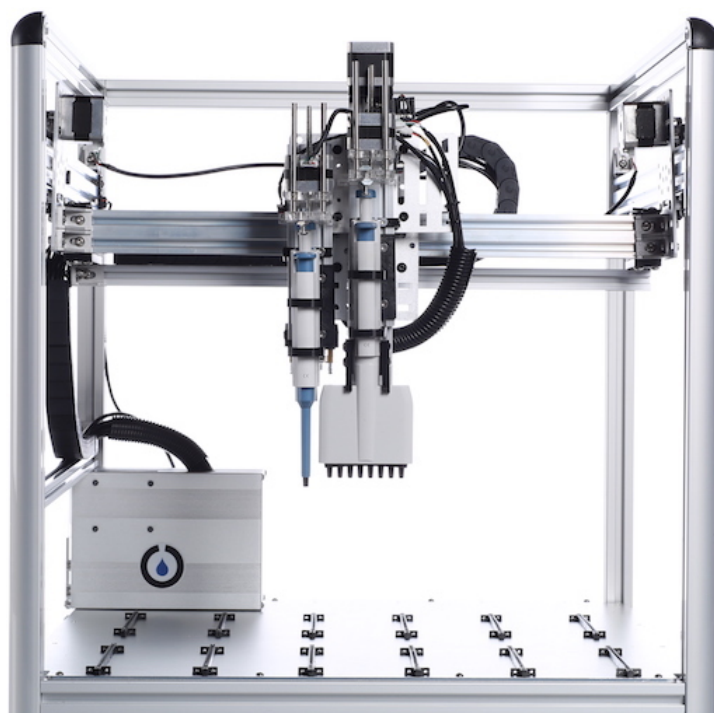


Figure 22: OpenTron an open source pipetting robot obtained from [79].

OpenTron is made of several stepper motors operating on a metallic framework, by controlling these the robot can steer in the X, Y and Z axis within its platform. The robot has metallic profiles which makes it easy to customize different arms for the robot, allowing it to perform other tasks than pipetting, such as air sampling. The user communicates with the robot through a built in graphical user interface (GUI) or through a Jupyter notebook. Custom protocols can either be uploaded in the GUI or used directly from the notebook. Thanks to developers all over the world contributing to the OpenTron platform, a python library allows custom protocols to be written. The community around the OpenTron is a great advantage as scientist all over the world contribute to developing and wiring protocols, sharing them on the OpenTron web site. In this thesis, the OpenTron platform is customized for continuous air sampling and a protocol for sampling is written.

### 2.1.2 Customization of the platform

Customization is necessary to adapt the OpenTron platform from pipetting to performing air sampling tasks. Customized parts to hold syringe and air tube has been design in fusion 360 and 3D printed (Figure 23). Vial boards are laser cut to reduce the vials from moving out of place during operation (Figure 23). The vial board supports the vials at the top and bottom. This minimalizes possible movement and helps to maintain a constant distance between the vial centres, which is important for the sampling algorithm due to the predefined path.

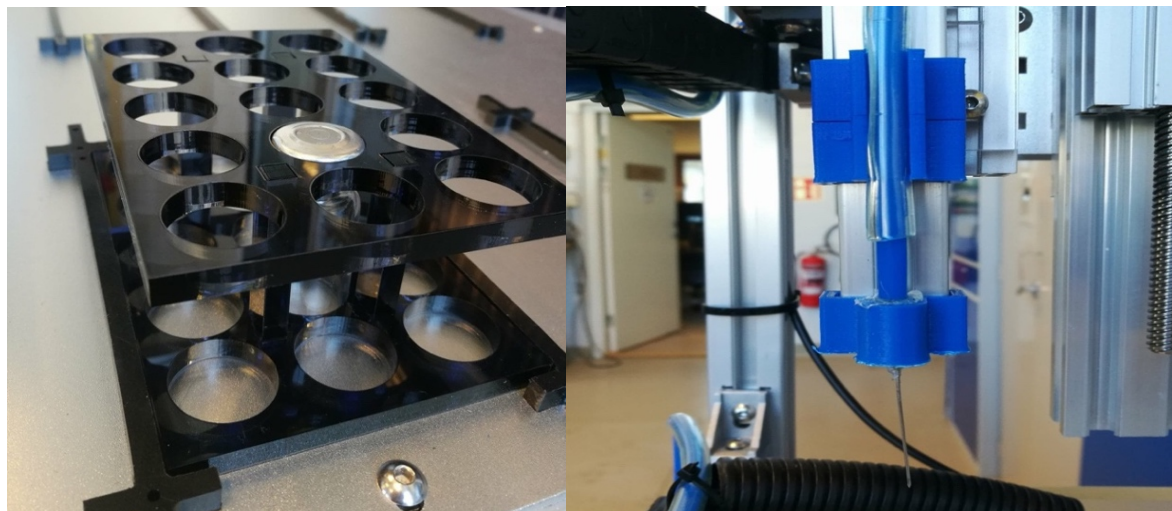


Figure 23: a) Vial holder that keeps the vials from moving during operation, b) 3D-printed air tube and syringe holder.

### 2.1.3 Algorithm

For automatic air sampling a distinct pattern of movement had to be defined. The needle must puncture the sealed part of the vial (Figure 24) and avoid puncturing the same spot twice, repeating the procedure through all the 15 deck slots with racks consisting of 15 vials in each rack. Time for purging, flushing the tubes, and filling the vials must also be set. Avoiding puncturing the same spot twice is important to elongate the lifetime of the vial, so that the same vial can be used for multiple samples.



Figure 24: The red circle represents the sealed part of the vial, a circle with a diameter of 5 mm.

Precision puncturing was done by creating a custom board and defining the puncture area, a rotation argument was utilized for avoidance of double puncturing. Time for purging is obtained from the field-test protocol and set as a pausing argument. The code flow chart can be seen in Appendix B: Code flow charts, and the source code is found in Appendix C: Source code and data. Error handling is also implemented. Loss of communication with the master (in this case the PC) causes the robot to stop, when connected again the robot must be reset. If an endpoint switch is hit unexpectedly the robot stops. After the robot finishes its operation, a .txt file is created describing all the operations performed by the robot. This can be used for debugging or double checking the samples.

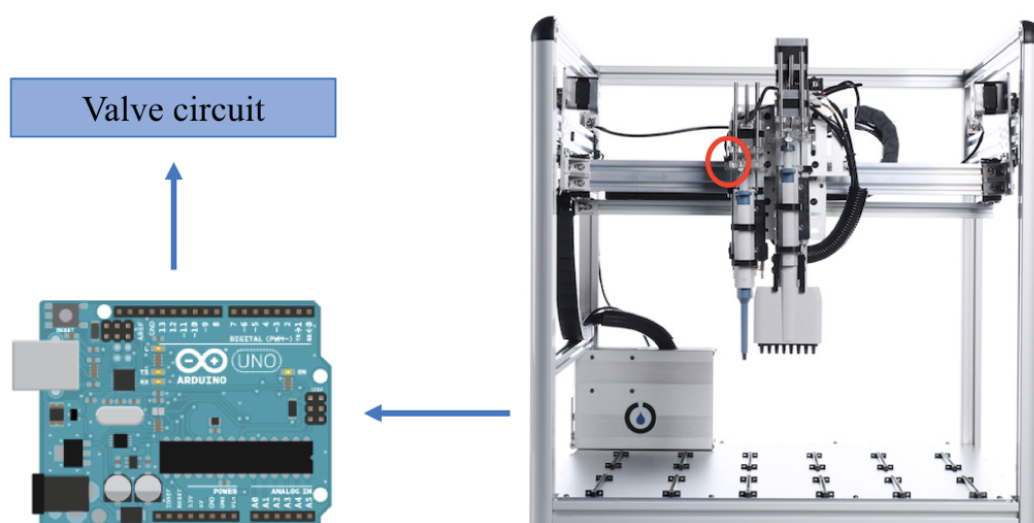


Figure 25: Signal flow (blue arrows) from the b switch circled in red to the valve circuit

#### 2.1.4 Operation

Operation of the OpenTron is done through a Jupyter notebook. Jupyter notebook is an IDE for different programming languages, where different code can be run independently from each other in cells. In the notebook different cells have different functions, the first cell connects to the robot over USB, the second cell homes the robot and the third cell contains the sampling script. When the third cell is run, the robot starts the sampling procedure. To run the cells one simply hits shift and enter. During operation print statements print what sample the robot took and if any errors has occurred. For in depth explanation se the operation manual in Appendix D: Operation manual.

For first time operation of the robot with a new script the first vial position and the syringe depth needs to be defined. This is done through the OpenTron GUI and is further explained in the operation manual found in Appendix D: Operation manual. 24-hour operation with no human interference requires a robust code and sources of error handling. Possible errors the OpenTron might face during operation are displacement error, jamming of the motors if the

robot moves out of bounds, needle malfunction and wrong sampling procedures. Handling these errors is not yet implemented.

### 2.1.5 Pump and valve circuit

A special valve circuit has been designed for pumping air and earth air into the tube of the autosampler. The valve circuit communicates with the OpenTron by reading its b switch, circled in red in Figure 25. By reading the signal from the switch a port is set high on the Arduino when the switch is pressed. This is accomplished by homing the b-axis on the OpenTron. If the Arduino reads the switch as high the air/earth valve is switched see attachment B.2 Valve and pump circuit code for flow chart of the Arduino code. By activating the switch after taking a sample, every other vial is thus filled with an air or an earth sample as seen in the code flow chart for the Arduino presented in attachment B.3 TakeSample function. The logging sequence is logged both from the Arduino and the OpenTron in a separate file if the operator becomes unsure of the sampling pattern.

### 2.1.6 Pump and valve circuit design

The electronic circuit consist of an Arduino Uno microcontroller, transistors, diodes and an external power supply. Arduino microcontrollers are small computers that can be programmed to send and receive electrical signals, allowing it to interact with electronic components. The transistor (type 2N2222) is used as a switch, when the current flows in the collector, the base and emitter is connected allowing power to flow. The diode (type 1N4007) only allows electricity to pass one way, functioning as a failsafe for leakage currents. An external power supply is used as source for the circuit as the Arduino cannot deliver the requested power so an external power supply that can be regulated is used. Small 3 volt pumps are used to pump air from the soil and outside to either the sensor or the OpenTron. Three different three-way valves running on 3 volts, determine the flow of air in the tubes. Either directing the air to the sensor or to the OpenTron seen in Figure 26 b). The valves and pumps are activated through a power control circuit seen in Figure 26 a).

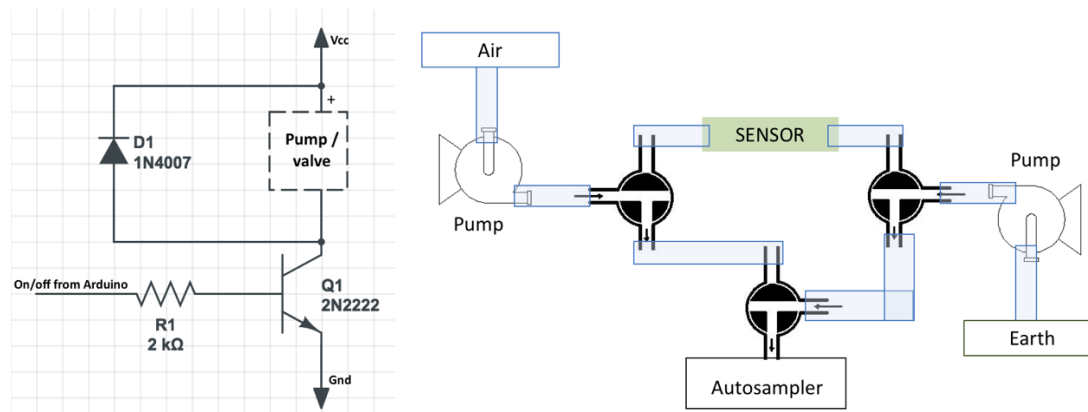


Figure 26: Showing the power control circuit in a) and the pump and valve setup in b)





### 2.1.7 Autosampler test

The autosampler setup was stress tested for 1 hour, sampling 15x15 vial samples. All samples were taken on the same rack to spare already vacuumed vials for real testing. The vial rack was covered with a paper sheet to reveal the circular sampling pattern of the robot arm.

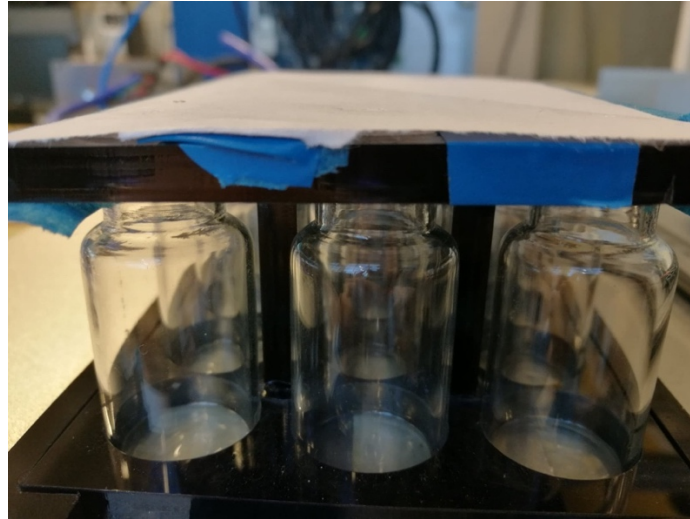


Figure 27: Test setup of the vial rack with the paper sheet on top

The pump circuit was also tested during the stress test. Due to stability problems with the three-way valves, caused by poor manufacturing of the valves, they were replaced with lights, showing which valve should have been activated. Figure 28 Shows the valve circuit, the yellow cord to the right reads the b-axis from the OpenTron. Signaling to the Arduino when to turn the sampling script on. The sampling script is presented in Appendix C: Source code and data.

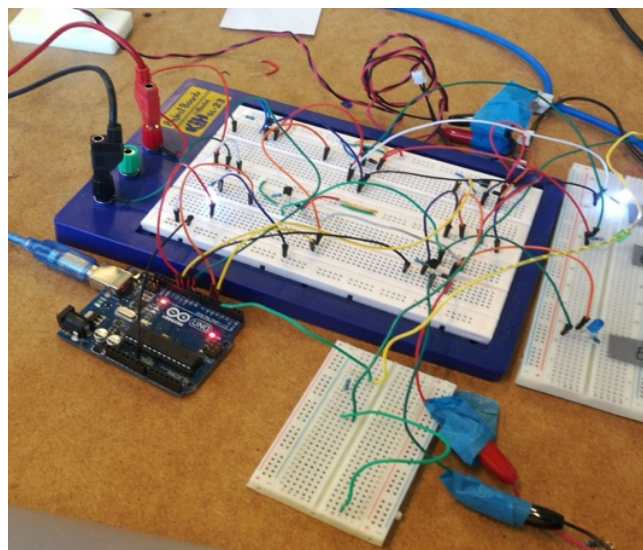


Figure 28: Pump and valve circuit set-up

## 2.2 Data collection

For long term testing of the CMUT-sensor it was set in a greenhouse for 4 weeks to obtain CO<sub>2</sub> measurements. The greenhouse environment was humid and the temperature varied from 25 – 33 °C depending on the outside temperature. During the greenhouse experiment two sources of data were collected; frequency shifts produced by the sensor array and reference samples for the GC. Ambient air is used as the base line for the samples providing a clear shift when the chemical is introduced and to make up for local changes in the ambient air. Soil, known to have a high concentration of CO<sub>2</sub> due to its high rate of microbial activity [80], is used as the test sample. Comparing the soil to the air sample the deviation between the samples results in the absolute CO<sub>2</sub> concentration in the soil bed. As the sensor outputs the frequency shifts the difference between the frequencies is given by;

$$\Delta f = f_{air} - f_{earth} \quad [5]$$

The gas chromatograph measures the concentration of CO<sub>2</sub> in ppm for the earth and air samples. Differences in concentration between the earth and air samples are used as reference for the frequency shifts. The frequency shifts are used as input features in the data analysis part. Appendix E: Greenhouse test protocol describes the data collection procedure in detail.

A mat lab script sets the resonance frequency for the CMUT sensors and logs the frequency shifts (Hz) over a time interval of 6 minutes. Using team viewer on the main test computer the sensor testing can be done automatically. Air samples for the GC are collected by pumping air into vacuumed sealed glass vials, collecting air samples requires a human operator, each sample takes about 10 minutes to complete.



**Figure 29:** Typical setup for a testbed, the bubble stone sparger is represented in blue and is placed beneath the soil surrounded by gravel to prevent the saturated soil clogging the air inlet.

Earth samples are obtained by placing a sparger surrounded by gravel beneath soil (Figure 29). Gravel prevents saturated soil from clogging the sparger inlet. Two different types of spargers

were used; “gas measuring tube” and a bubble stone (Figure 30), at different heights. The “gas measuring tube” is a tool developed for measuring gas in soil. Altering sparger types and their height leads to different CO<sub>2</sub> concentrations. Soil irrigation was also varied to change the composition of the gas mixture. A pump is used to pump air from the soil over the sensor or into a glass vial.



Figure 30: Different sparger types, a) ”gas measuring tube” b) bubble stone sparger

### 2.2.1 THE CMUT CHEMICAL SENSOR

The CMUT chemical sensor used, is developed by Dr. Quintin Stedman [9] and consist of an array of 6 sensors with different polymer coatings and one sensor without coating for reference. Acting as a general-purpose set of functionalization materials, the polymer coated array can be used for a variety of chemical sensing experiments.

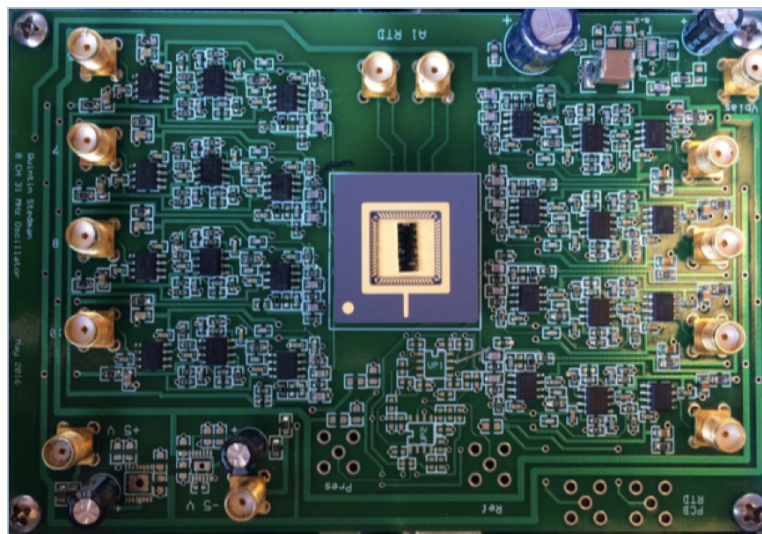
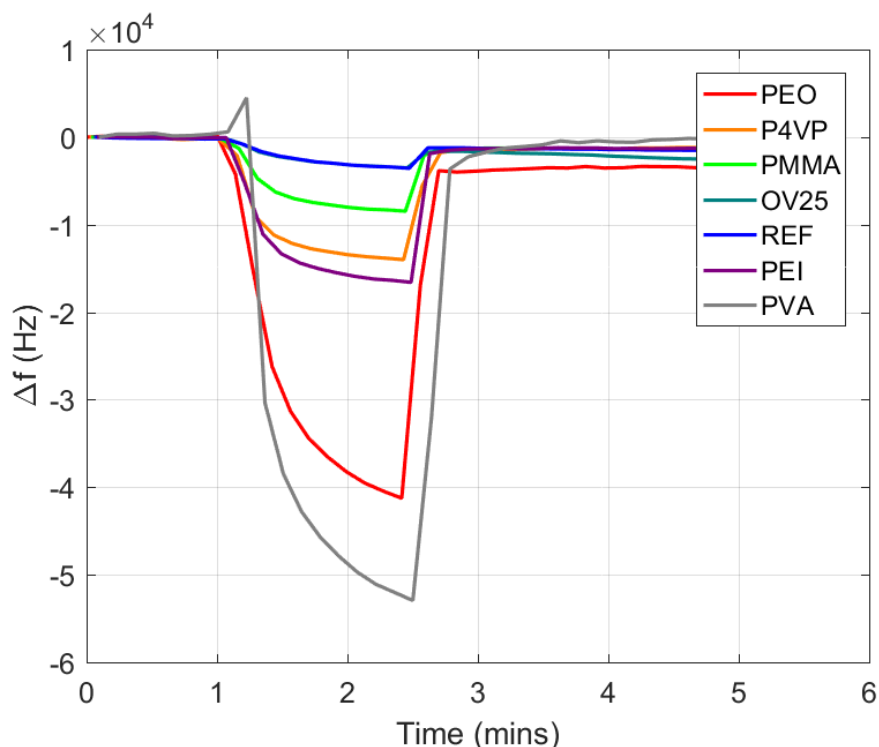


Figure 31: The CMUT Chemical sensor with circuit board for determining frequency

The different polymers used in the field test and their sensitivities are listed in Table 2. As seen in Figure 32 each sensor responds to the sample but each sensor provides a unique frequency response. The frequency responses are used as features for the machine learning models.

**Table 2: Polymers used as coatings for the CMUT sensors during the experiment and their sensitivities**

Polymer	Sensitivity
PEO (Polyethylene oxide)	Water, acetone, methanol, ethanol, isopropanol.
P4VP (Poly(4-vinylphenol))	Acetone, methanol, ethanol, isopropanol.
PEI (Polyethylenimine)	CO <sub>2</sub> , some N <sub>2</sub> O sensitivity, low water sensitivity.
OV-25(OhioValley#25, Phenylmethyldiphenylsilicone)	Good selectivity to isopropanol and acetone over water.
PMMA (Polymethyl methacrylate)	Sensitive to acetone and ethanol. Low sensitivity to water vapour.
PVA (Polyvinyl alcohol)	Water vapour
REF (Reference)	No polymer coating, used as reference



**Figure 32: Sensor response to CO<sub>2</sub>, showing that each functionalization layer have different responses**

## 2.2.2 Gas chromatography

To collect reference samples for the sensor gas chromatography (GC) is used. GC identifies volatile substances in the gas phase [81]. During the test the samples were analysed at NMBU in collaboration with Dr. Peter Dorsch to detect relevant gases, in this case CO<sub>2</sub>. The working principle of a GC is shown in Figure 33. Here, a desired sample is injected through a port and is transported through the column by a carrier gas. A carrier gas is a gas that is not chemically reactive such as nitrogen, helium, argon and carbon dioxide and is used to remove impurities, remove water and for transportation [82]. Inside the column, the compounds are heated and separated. The column consists of a long tube that is coiled with some liquid coating on the tube walls. By attracting some compounds whilst other compounds are repelled the liquid coating enables the separation. The separation is dependent on the boiling points of the inserted compounds. A compound with a high boiling point is more likely to stick to the sides of the column, while a compound with low boiling point would most likely go into gas phase rushing through the column without touching the walls. When the compounds have travelled through the column they hit a detector. The detector measures the intensity of each compound (how many particles are hitting it) over time. This is shown in the chromatogram with peaks representing the intensity of each compound. The first peaks in the chromatograms is the compound with low boiling point whilst the later peaks are compounds with high boiling points. The relative difference between the peaks gives us information on what compounds our sample is made of. To get a more precise reading the sample is also run through a mass spectrometer.

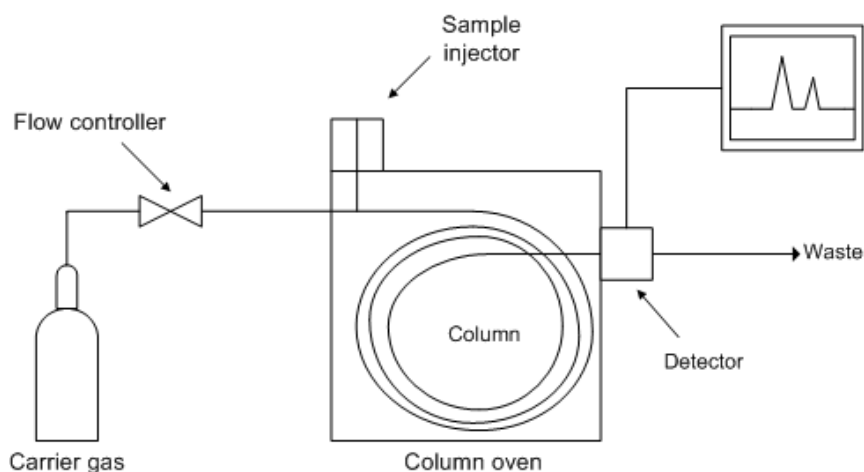


Figure 33: Simple drawing of a gas chromatograph from [83]

During the sensing experiment air samples were analysed in a gas chromatography providing a chromatogram. Through analysis in excel concentration and chemicals present are obtained used as reference for the sensor array.

### 2.2.3 Data analysis

During the data analysis, data produced by the sensor is used as features and the GC results are used as reference values. Both are inputs to the machine learning models. Figure 34 describes the main steps; input data are pre-processed before the data is given to a classification or quantification model. The model outputs the belonging class or concentration of a given sample, in Figure 34 either a high or low concentration.

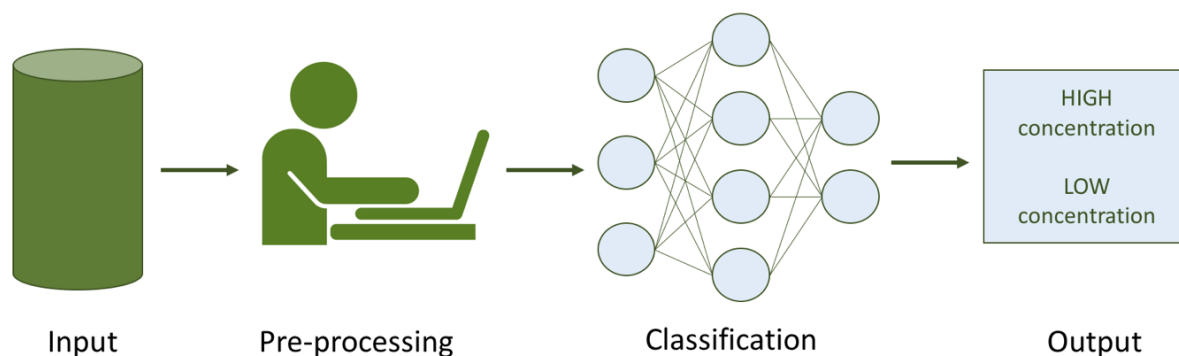


Figure 34: Steps in machine learning

In the classification step machine learning was used to classify chemicals and its concentration based on the frequency shift produced by the sensor array. Measurements are divided into testing and training sets which was fed into a machine learning model. The prediction is cross-validated and a score based on the output is given used as a comparison factor for the different models.

## 2.2.4 DATASET DEVELOPMENT

197 samples from the greenhouse test were gathered winter 2017 presented in Appendix C: Source code and data, and used as input for the machine learning models. In the greenhouse, the sensor array was used to identify CO<sub>2</sub> at different concentrations in a testbed. After the sensor samples were gathered data was processed at Stanford University, while data processing of the GC results was done at NMBU. The results were gathered in a comma separated value file. The data shows what week the samples were taken in, sample numbers, concentration of earth, air and their difference according to the gas chromatography, the frequency shifts from the CMUT and the time since start of the set. Features used as input for the machine learning algorithm are summed up in Table 3.

**Table 3: Features and reference used as input to the machine learning models**

Feature	Type of data
PEO	Sensor response
P4VP	Sensor response
PMMA	Sensor response
OV25	Sensor response
REF	Sensor response
PEI	Sensor response
PVA	Sensor response
Classification (used as reference)	Concentration over 350 ppm = High (1) Concentration below 350 ppm = Low (0)

## 2.2.5 SOFTWARE

Data analysis is performed with Python3 and some of parts of the standard libraries; *scikit-learn*, *numpy*, *pandas* and *matplotlib* [84-87].



### 2.2.6 Data preparation

Data preparation is an important step in machine learning, the foundation of the model prediction is set through selecting, cleaning and transforming the data. First the desired data needs to be selected and cleaned. Selecting and cleaning the data consist of transforming the file format, selecting the data with the most useful information (feature selection) removing or recovering missing data or incomplete samples. Secondly the data is transformed. Transforming the attributes of the data to a standard Gaussian distribution with a mean of 0 and a standard deviation of 1 makes algorithms like linear regression and logistic regression perform better because all attributes are weighted evenly.

Data preparation is often an iterative process as different types of transforms exposes different structures of the problem. Due to the small number of features present in the dataset few iterations were necessary. The preparation steps are implemented before the machine learning models were trained. Summed up in Table 4.

**Table 4: Data preparation steps; data hold back, standardization and row removal**

Method	Description
Data hold back	Holding back to measurements for final objective test of algorithm
Standardization	Removing mean and scaling to unit variance
Row removal	Removing rows with two or more missing variables

## 2.2.7 MACHINE LEARNING MODELS

Algorithms tested are the logistic regression, SVM with linear, rbf and polynomial kernel as well as a neural network. The algorithms are scored by leave one out cross validation and accuracy. The data was classified before quantification by regression was performed.

For all machine learning models the data is split up into random *testing and training subsets* to test the models. `sklearn.model_selection.train_test_split` [88] was used to split the data into testing and training sets. 40% of the data is held back for testing purposes. Splitting the data also avoids overfitting models by forcing generalization on the training data.

### 2.2.7.1 Classification algorithms

The main goal of the classification algorithms is to classify a sample as either a high or a low concentration of CO<sub>2</sub>. For classification purposes attributes that are over 350 ppm are classified as high and set to equal 1. Those lower than 350 ppm are set to 0, classified as low.

*Support vector machines* (SVM) requires a Gaussian distribution about 0 and data that is within the standard range of (0,1) or (-1,1) so that all variables are treated equally. The scaling is done with the `sklearn.preprocessing.StandardScaler` [89] and the SVM is performed with the `sklearn.SVM.SVC` [90], The different kernels used are: polynomial, linear and radial based.

By tuning the  $C$  and  $\gamma$  parameters, modification of boundaries was obtained. The effect each parameter has on the boundary is summarized in Table 5. To find the value of  $C$  and  $\gamma$  that provides the best result a grid search is performed. The grid search is done using the `sklearn.grid_search.GridSearchCV()` [91] function. `GridSearchCV` performs a search of specified values for  $C$  and  $\gamma$ . The algorithm returns the parameters with the highest testing and training scores after cross validation. This is time consuming and should only be done once. By using grid search,  $C$  and  $\gamma$  were found presented in Table 6.

**Table 5: Information summary of  $C$  and  $\gamma$  parameter**

Parameter	Info
$C$	Penalty parameter or the error term, acts as regularization parameter. Controls the trade-off between a smooth decision boundary and classifying the points correctly. <i>Low values</i> indicates a larger hyperplane and allows for more misclassifications [92].
$\gamma$	Kernel coefficient for 'rbf', 'poly' and 'sigmoid' kernels. Gamma influences where the decision boundary is set. <i>High values</i> of gamma indicate that points close to the decision boundary are heavily weighted, resulting in a wiggly boundary. <i>Low values</i> of gamma indicate that points that are further away from the decision boundary are heavier weighted, resulting in a straight decision boundary [92].

*Logistic regression* is performed with `Sklearn.linear_model.LogisticRegression`, and parameter  $C$  is tuned for the best fit. In logistic regression  $C$  gives an indication of how much we want to avoid misclassification.  $C$  can also be found by using grid search, this was implemented. The value the grid search put forth was used as a starting point, little tuning was done to find the best parameter of  $C$ .  $C$  was set to 20.

*A neural network* is set up using `sklearn.neural_network.MLPClassifier` which creates a multilayer perceptron classifier. MLP-classifier needs to have a definition of the hidden layer size activation and solver. The hidden layer size represents the number of neurons in the hidden layer, activation defines an activation function for the hidden layer this is set to `relu` which is the rectified linear unit function while the solver is set to `lbfgs` which is an optimizer that returns  $f(x) = \max(0, x)$ .

**Table 6: Summaries of machine learning models for classifications used and final setting for their parameters**

Algorithm	Parameters
Neural Network	8 layers deep
Logistic regression	$C = 20$
SVM	Kernel: Gaussian $C = 10$ , $\gamma = 1$ Kernel: Polynomial $C = 1$ , $\gamma = 0.2$ Kernel: Linear $\gamma = 1$ , $C = 10$

All algorithms were scored by a cross validation, accuracy and a confusion matrix was made from the predictions. For a closer detail see the machine learning pipeline code shown in Appendix C: Source code and data.

### 2.2.7.2 Quantification algorithms

The goal of the quantification algorithms is to correlate the frequency shift with the  $\text{CO}_2$  concentration present, based on linear relationships.

*Regression* was performed with `Sklearn.linear_model.LinearRegression()`, default settings were used.

*Lasso regression* was done with `Sklearn.linear_model.Lasso()` parameters altered was `alpha`, `fit_intercept`, `normalize` and `max_iter`. The `alpha` value is a constant that multiplies with the L1 term equivalent to an ordinary least square. The `fit_intercept` is set to `false` and no intercept is used in calculations, this can only be done if the data is centred. `normalize` is set to `false` as the data has already been standardized. `Max_iter` is set to 100000 and is the maximum number of iterations performed by the regularizer. Optimization objective for the lasso function is done by adding a penalty equivalent to the square of magnitude of coefficients.



*Ridge regression* was performed with `Sklearn.linear_model.Ridge()` with alpha set to 0.003. Alpha is the regularization strength; large values results in stronger regularization and adds a penalty equivalent to absolute value of the magnitude of coefficients.

## 2.3 Prediction

When the machine learning algorithm has processed the data, it returns a prediction and a score. To be able to tune and evaluate the algorithms, scores are evaluated and further tuning is implemented to investigate the possibility of improving the score. After the algorithm has been tuned and the score is satisfactory a final model is decided on.

### 2.3.1 Evaluation and scoring metrics

Metrics are used to evaluate machine learning algorithms by looking at the accuracy and precision of the algorithms prediction. For the classification problems the classification accuracy, logarithmic loss and confusion matrix are used to evaluate the algorithms. And for the regression mean absolute error, mean squared error and R-squared are used to evaluate the algorithms.

## 2.4 Results

In this section, results from the chemical sensing experiment and autosampler test is put forth. 197 samples were collected during the 4 weeks, 67 samples were removed due to human error. This left 130 reference samples for air and earth and 130 frequency samples from the sensor, for further investigation. From the 130 samples 2 samples were taken out at random to get a final evaluation of how well the machine learning models predicts. All graphs and plots are produced with the PyPlot library in python. As an added value to the chemical sensor the abilities of the autosampler was tested. Sampling 15 vials 15 times on the same rack within an hour.

### 2.4.1 Data visualisation

The Gas chromatograph measurements, reference samples for the earth, air and their difference, are presented in Figure 35. As seen in Figure 35 the CO<sub>2</sub> concentration in the earth is higher than the concentration in the air. This is expected due to the nature of soil as explained in section 2.2. The difference between the concentrations, air and earth, is the bottom line, used as the reference for the machine learning algorithm.

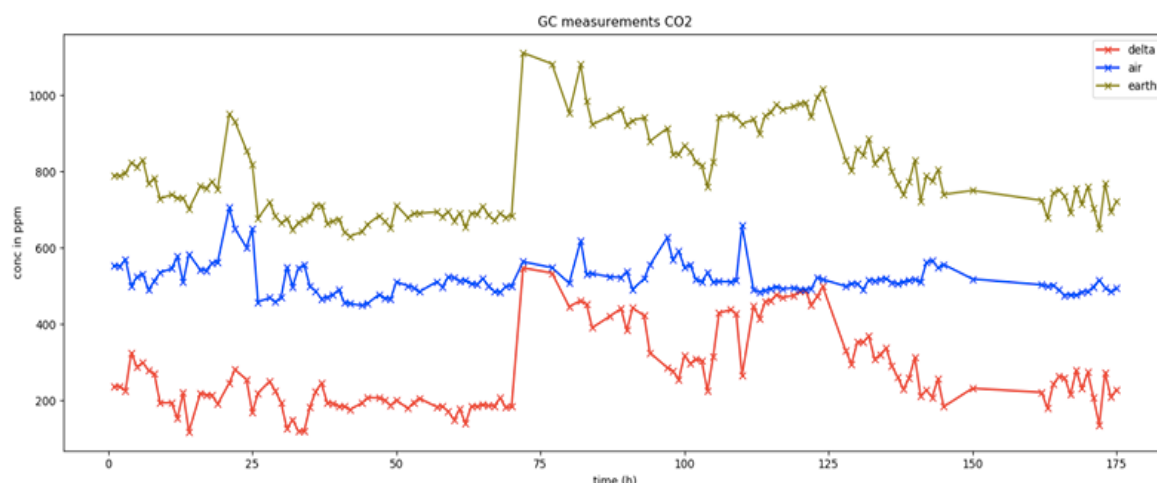
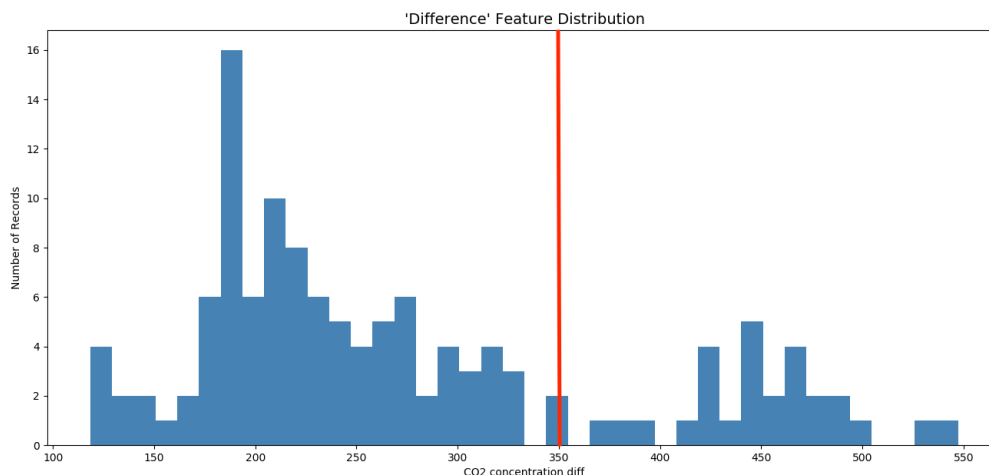


Figure 35: Gas chromatography results. Earth is presented in green, air in blue and their difference in red.

Distribution of the reference samples in absolute values are shown in Figure 36. The redline indicates 350 ppm, which is used as a cut off limit for the classification. High values are those above 350 ppm, and low values are those below 350 ppm. As seen in Figure 36 most values are below 350 ppm. Few values are higher than 350 ppm. When most values are concentrated in one class the data is said to be unbalanced.



**Figure 36: Data Distribution of the difference features used as reference for the frequency shifts, a concentration among low values are seen**

### 2.4.2 Classification

The difference between the earth and the air concentration of CO<sub>2</sub> was used as reference for training the machine learning models. The first task for the machine learning models were to classify samples as either high or low depending on their concentration. All concentrations above 350 ppm were marked as high while all below 350 ppm were marked as low. The collected data was split into testing and training sets. The models were trained on the training data and tested on the test data. Cross validation and accuracy scores were put forth. In Table 7 the different accuracy scores of the machine learning models are presented. Test size was 40% of the total training set, all models were trained on 60% of the data.

**Table 7: Accuracy and cross validation scores of the different machine learning algorithms, logistic regression, linear SVM, Gaussian SVM and neural networks.**

Algorithm	Time (s)	Cross validation score	Test Accuracy
Logistic regression (C = 20)	0.24	0.84	0.83
Linear SVM (C = 1)	0.24	0.67	0.68
Gaussian SVM ( $\gamma = 0.5, C=1$ )	0.25	0.87	0.84
Polynomial SVM ( $\gamma = 0.2, C=1$ )	0.23	0.88	0.88
Neural Network 8 hidden layers, relu solver	0.25	0.56	0.85
Neural Network 8 hidden layers, logistic solver	0.26	0.77	0.85



Model scores presented in Table 7 shows high scores for all models. It is apparent that all models use about the same time to put forth a prediction based on the presented data. With a score of  $0.88$  for both accuracy and cross validation the *SVM with a polynomial kernel* has the best performance of the models. Other models also prove to have a high accuracy, except for the neural network with a relu solver. A large deviation between the cross-validation score and test accuracy is seen in the neural network which is an indication of overfitting the model.

Although giving an indication of performance, it is important to keep in mind that the accuracy is the number of correct predictions made divided by the total number of predictions. When applied on unbalanced data, accuracy scores will also be unbalanced. The cross validation uses the entire dataset and cross validates the predictions, but is also affected by the unbalanced data. The best indication of model performance is thus the confusion matrix, which presents an overview of how the model guesses on the different data.

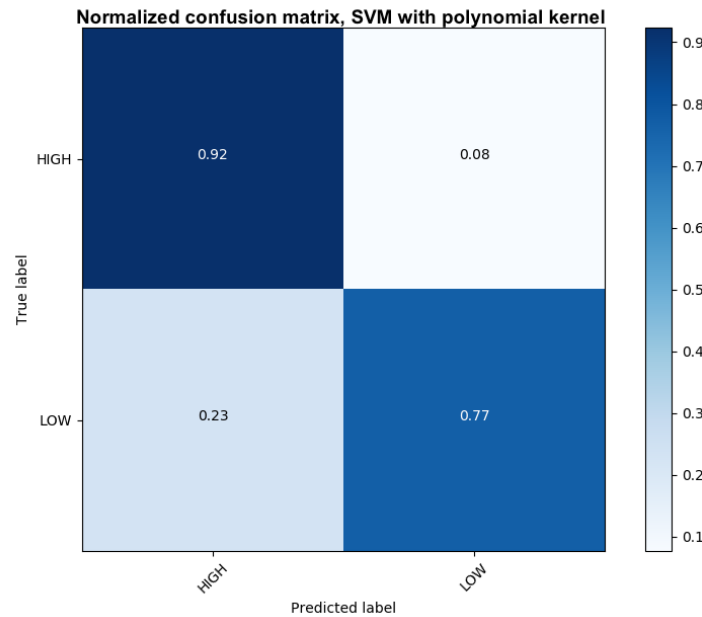


Figure 37: Confusion matrix for a SVM model with a polynomial kernel.

As seen in Figure 37, the SVM with a polynomial kernel guesses right most of the time. A high percentage of false positives is observed. The model guesses high on 23% of the data that is classified as low. Guessing low 8% of the time when the concentration is classified as high, the model seems to perform well on the concentrations classified as high.

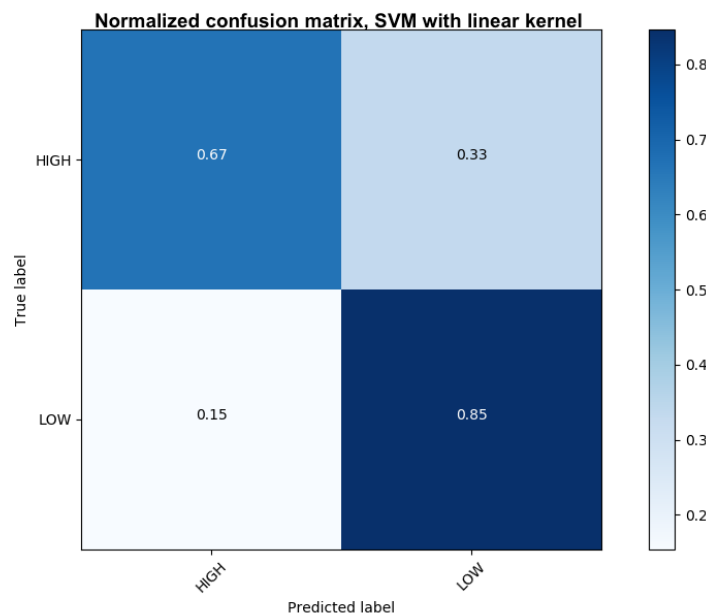
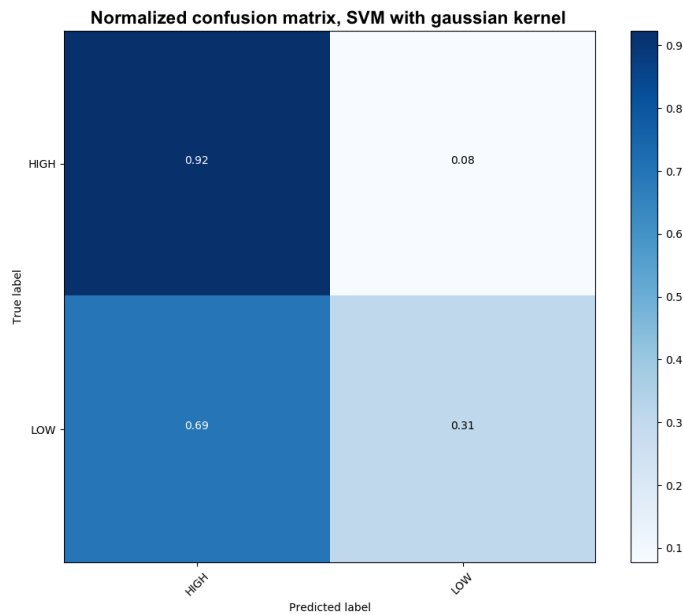


Figure 38: Confusion matrix for a SVM model with a linear kernel

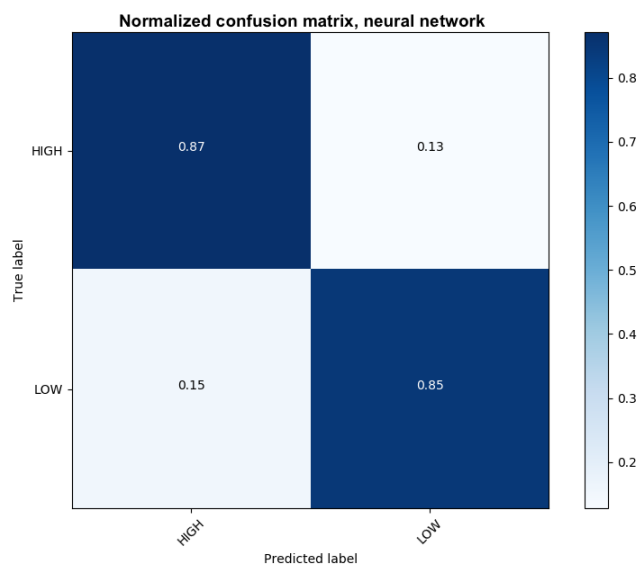
Figure 38 shows that the linear kernel has a high rate of false negatives, 33%, and a low rate of false positives, 15%. The SVM model with a linear kernel shows better performance on the concentrations classified as low, and shows a worse performance is shown on the concentrations classified as high.





**Figure 39: Confusion matrix for a SVM model with gaussian kernel**

Figure 39, shows a good performance of the SVM model with a Gaussian kernel on data classified as high. Worse performance is seen on the concentrations classified as low, as the model predicts high on 69% of the data classified as low.



**Figure 40: Confusion matrix for the neural network with RELU activation and 2x2 layers**

Figure 40 presents the confusion matrix for the neural network with relu activation and 2x2 layers. The model has an even prediction for low and high values, guessing wrong on 13% on the high data and 15% on the low data. Showing the most even performance of all models.

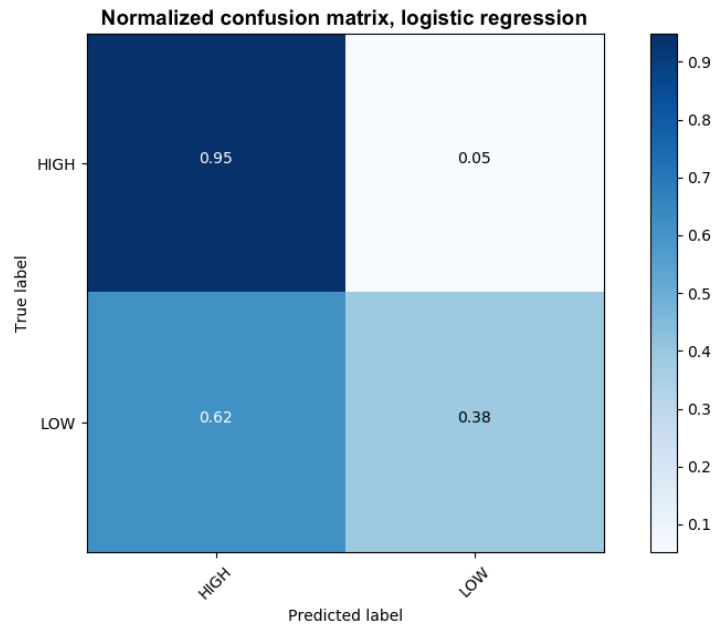


Figure 41: Confusion matrix for the logistic regression model

Figure 41 shows that the logistic regression model guesses good on data classified as high only guessing wrong on 5%. Logistic regression guesses wrong on 62% of the data classified as low, showing poor performance.

After all the models were tuned they were tested on the two randomly selected samples. The samples had a reference of 228 and 337 ppm with corresponding frequency shifts, and should default be classified as low. All models classified the samples right, which is expected as the chances of classifying a random sample from the set as low and being right is 77%.

### 2.4.3 Quantification

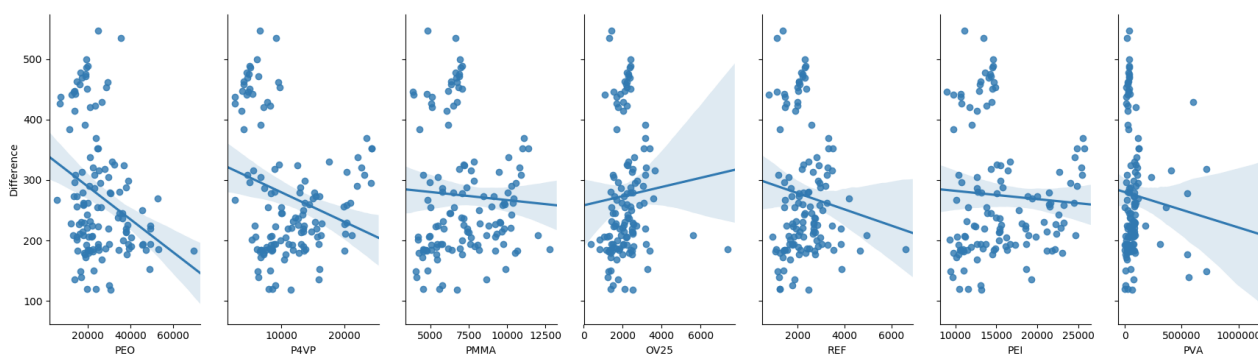
To map the input, frequency, to the reference, concentration. Linear, lasso and ridge regression models were used. The main difference between the models are their error estimates. Parameters are set in the methods chapter and the model errors are shown in Table 8. The models were tested on 40% of the data set, and trained on 60% of the data. Test and train scores are the R-squared score for the testing and training set.

**Table 8: Regression models and their scores entire data set**

Model	RMSE	Test score	Train score
Linear regression	98,3	0,23	0,28
Lasso regression	98,4	0,22	0,28
Ridge regression	98,3	0,23	0,28

An interesting result is the similarities of the train and test scores. The scores deviate about 0,05 from each other indicating that the all the models generalize well to the data and overfitting has been avoided. Another interesting observation is that the linear and ridge regression models produce the same scores. Indicating that the best fit of the data is indeed a straight line, and that all features adds value to the prediction.

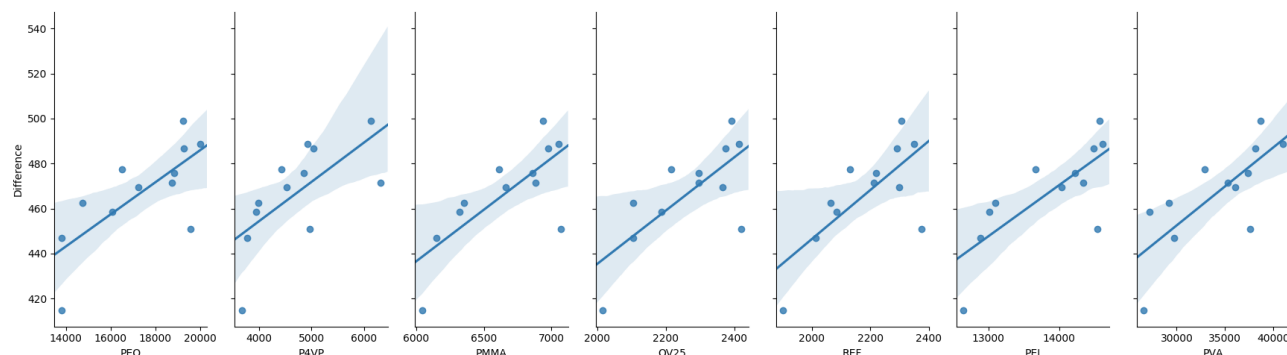
Despite the good generalization, the models show poor performance of predicting well on the data. High RMSE scores indicate poor performance. A possible indication of the bad performance is seen in Figure 42. Figure 42 shows each feature mapped independently and a linear model fitted to the points. There is no good linear pattern in the points, therefore the data was split up into different sampling periods.



**Figure 42: Scatter plot and the regression line with 95% confidence interval of the CO<sub>2</sub> difference and the different frequency responses for the entire dataset**

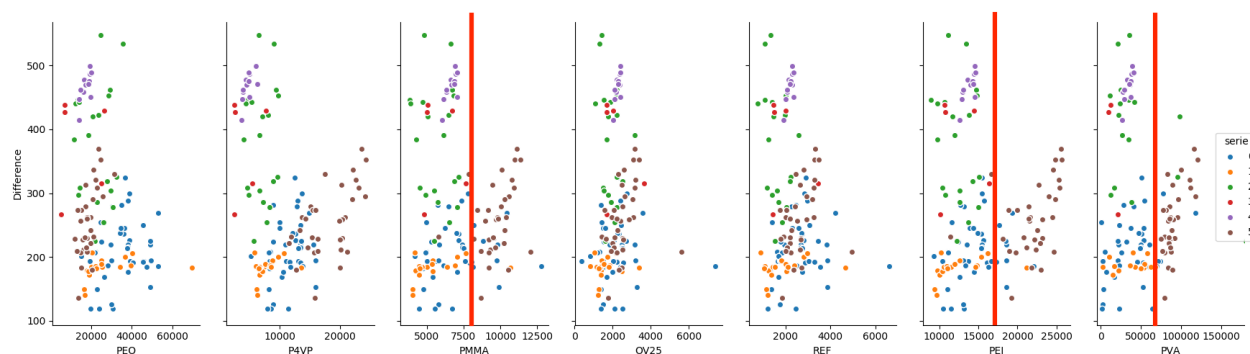
Figure 43 demonstrates the linearity found in the first period. A linear pattern is in fact seen, concentration increases with the frequency for all polymers. Although a linear pattern is found, it is important to note that the problem is multidimensional and patterns might be found when

looking at the seven-dimensional problem. The two-dimensional plots only give some indication of what causes poor performance.



**Figure 43: Scatter plot and the regression line with 95% confidence interval of the CO<sub>2</sub> difference and the different frequency responses for week 45-46 with the gas measuring stick**

To further investigate the poor performance different samples for different sampling periods were plotted in different colours in Figure 44. Each colour represents a sampling serie 4 is when the gas measuring stick was used others use the bubble stone sparger and follow a weekly basis. For the PMMA, PEI and the PVA it is evident that the responses at the same concentrations are shifted higher up at the end of the sampling period, week 4 and 5. It is also seen that high concentrations are all taken in the same period.



**Figure 44: Sensor responses plotted in a different colour for different sampling procedures.**

A shift in response is made more evident when looking at the sensor responses. Figure 45 a) shows the sensor response in week 1 and Figure 45 b) shows the response at the end of the 4 weeks. A large increase in response is seen in the PVA functionalized sensor.

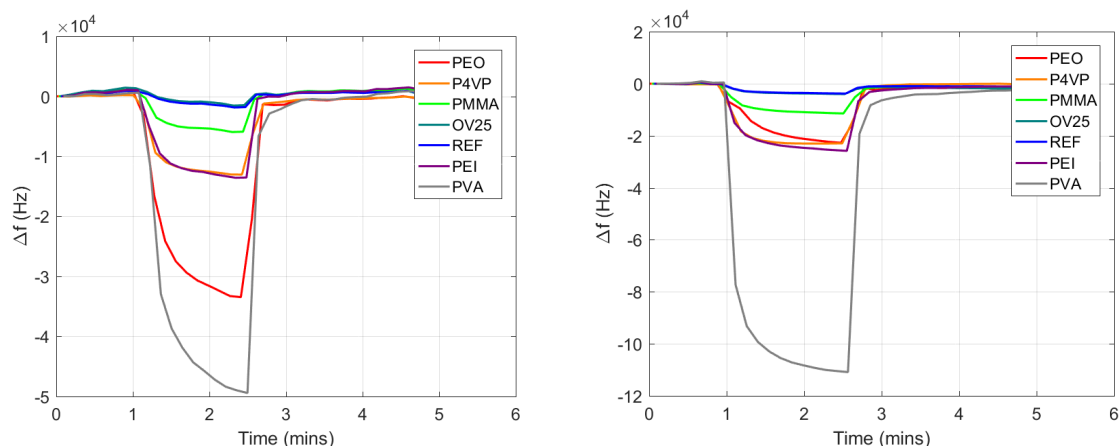


Figure 45: The difference in sensor response from the first week to the last week of testing. a) week 1 of testing b) after 4 weeks of testing

### 2.4.4 Testing the regression model on unseen data

After all the models were tuned they were tested on the two randomly selected samples. The samples had a reference of 228 and 337 ppm with corresponding frequency shifts. Table 9 summarizes the predictions. The Lasso model provides the best predictions having the best estimate for both samples. All models guess better on the 288 ppm sample which is the lowest of the two samples. When running the 337 ppm sample all models guesses a lower number. Indicating that the model has better performance on lower concentrations.

Table 9: Linear, lasso and ridge predictions on left out data

Value	Linear prediction	Lasso prediction	Ridge prediction
Value 1 = 288ppm	225 ppm	235 ppm	225 ppm
Value 2 = 337ppm	224 ppm	238 ppm	224 ppm

### 2.4.5 Autosampler test

The autosampler was stress tested for 1 hour sampling the 15 vials 15 times. No mistakes were done and the robot successfully sampled the vials (Figure 46). The precision of the autosampler was shown as the arm created a circular puncturing pattern with a diameter of 2 mm, ensuring that no vial was punctured in the same spot twice. Eight punctures can be seen in Figure 46 b), the robot followed the exact same pattern for the remaining seven samples with no offset. It is important to note the problems that occurred with the three-way valves during early prototyping of the circuit. As they were proven to be unstable and should be changed.

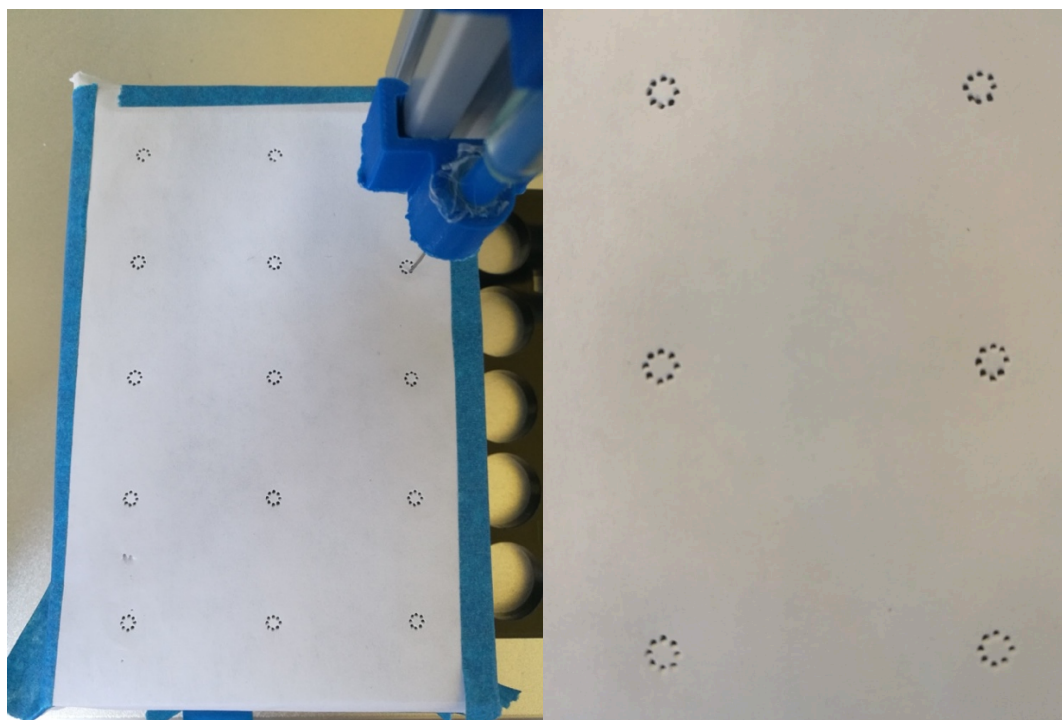


Figure 46: Showing results from the sampling test a) robot arm with syringe taking a sample b) distinct sampling pattern so that the same spot is not punctured twice

### 2.4.6 Summary of results

Presented in the results chapter is the unbalanced distribution of the sampled data, how well the classification models predicted on the data as well as the quantification models predictions, and the results of the autosampler test. Classification models were diverse in their predictions, the SVM models with polynomial and Gaussian kernels as well as the logistic regression model classified better on high data, whilst the SVM model with a linear kernel classified best on low concentrations. The neural network with a relu activation classified most evenly between the two. Quantification through regression showed more similar results. The models had large RMSE due to shift is observed in the data, indicating poor performance. All regression models predicted better on lower data than high. The autosampler was successfully tested on 15x15 samples creating a circular pattern for avoidance of sampling the same spot twice. Proving itself to be a valuable sampling tool.



## 3 DISCUSSION

Over the course of the chemical sensing experiment, the CMUT sensor responded appropriately to concentrations of CO<sub>2</sub> and had a reliable operation for a time-period of 4 weeks in humid conditions. The machine learning models were successfully implemented and provided good predictions for the classification task. However, skewed data and sensor drift led to poor predictions for the quantification task. Because machine learning models are only as good as the data they are trained on, the dataset is paramount to the success of the machine learning model. This will be an important topic in this discussion. Further, the chemical sensing experiment is discussed: how does it compare to other experiments? And how well does it reveal the capabilities of the chemical CMUT sensor? What are the end-user requirements and how does the sensor compare to other available sensors? At last the autosampler is discussed to identify improvements and shortcomings.

### 3.1 A model is only as good as the data its trained on

Because the machine learning model's prediction is based on the initial data set, the quality of that initial data is important. Of the total 179 samples were collected, 49 samples were removed due to human error during the sampling. Leaving 130 points as input to the machine learning models. The low amount of data sampled is a result of the labour-intensive nature of collecting the samples and analysing the GC data. Multiple research teams use the same GC lab which results in queues for analysing samples. Building models on 130 samples might lead to overfitting. When too few data points are obtained, little generalization is achieved and the model has a higher chance of presenting poor predictions. Consequently, a more accurate model would require more data. Data is the key ingredient of a good machine learning model, it can be thought of as a limiting factor for good performance.

A larger range of CO<sub>2</sub> concentration should be gathered as imbalanced data lead to poor performance of the machine learning models. With imbalanced data, the accuracy paradox comes into play. A 78% accuracy will be obtained if a classification model always predicts low concentration on any given data from the dataset used in this thesis. In this case, the accuracy score only reflects the underlying class distribution. As a result, the accuracy scores obtained from the models, are not a good reflection of the models performance. Machine learning models discover patterns in the data, when there are a lot of data distributed on low concentration the model weighs this heavily, and adjusts better to the data of low concentration. As a result, the models are good at predicting low concentrations, but show poorer performance at predicting high concentrations. An example of this is seen when the two left out samples are tested on the regression models, giving better predictions on the low concentration (288 ppm) than the sample with a higher concentration (337 ppm). Although unbalanced the classification models showed an ability to classify the samples with low error. With an error of only 13 and 15 % error, for the neural network and an error of 23 and 8% for the SVM model with a



Gaussian kernel, the models showed best performance of all the classification models. Comparing the results to the requirements set by DARPA which have a limit  $10^{-5}$  the models have poor performance with a limit of  $10^{-1}$ . Poor performance might be a result of overfitting the models. Failing to describe a generalization of the underlying pattern. Thus, failing to meet the requirements set by DARPA. To improve the models, a larger range of concentrations should be obtained as it helps exposes underlying phenomena and contribute to balanced machine learning models.

Drift due to polymer degradation was observed and identified as a reason for poor performance of the machine learning models. Due to the drift, no clear linear relationship is found between concentration and frequency. One explanation to the drift observed is the degradation on the polymers making up the functionalization layers. Polymers degrade when they are influenced by factors such as heat, light, vapour or other chemicals. The greenhouse was both hot and humid which explains the degradation process. To correct for the drift caused by polymer degradation, Di Carlo and Falasconi[93] suggest using orthogonal signal correction, fast Fourier transform or discrete wavelet transform. Investigation of these methods, as well as further research on sensing layers and the effects of humidity, is recommended.

### 3.2 Assessing sensor behaviour during chemical sensing experiment

The test performed in the greenhouse puts the sensor in a real-world environment where a mixture of substances is present. To evaluate the CMUT sensor the sensor is compared to a standard developed for chemical sensors in the Netherlands [94]. The standard presents requirements for chemical sensors used in a real-world environment.

One of the requirements set by the standard is that the sensor can detect composition changes in the ambient air, specifically gaseous emissions that pose a risk of odour nuisance and/or safety risks. The results from the greenhouse test show that the sensor responds to low concentrations of  $\text{CO}_2$  in ambient air. For this reason, one can assume that if the sensor is able to measure  $\text{CO}_2$  concentration in ambient air, it can also measure other chemicals at comparably low concentrations, consequently reaching one of the standard's requirements.

Further, the standard suggests that the local ambient air should be measured at different humidity rates, and that the chemical sensor should pose a warning if abnormalities are found. Recording the baseline of air will increase the prediction accuracy of the sensor as local natural changes to the air composition is taken into consideration. By measuring the local changes to the ambient air and looking for abnormalities with one sensor, another sensor could be activated when such an anomaly is found.

The CMUT sensor tested in the greenhouse was exposed to a high humidity and the ambient air was used as a reference. Flushing the CMUT sensor with ambient air gave a frequency shift equalling the absolute concentration of  $\text{CO}_2$  in the plant bed. When the CMUT sensor is placed

out in the field one may not be able to perform such a flushing routine. If this is the case the ambient air, at different humidity rates, should be recorded to correct the frequency shifts to the changes in surrounding air, as suggested by the standard.

Being able to recalibrate the system through periodic retraining is another requirement set by the standard. The ability of recalibrating allows the sensors in the field to adapt to changes such as drift and aging. Being able to adapt to these changes would lower the need of maintenance and elongate the lifetime of the sensors. Implementing a recalibration procedure for the CMUT sensor would require testing to find the behaviour over time for the different functional layers. When this is known, recalibration processes can be developed. Recalibration can be performed in the software eliminating the need of adjusting the sensors.

Differentiating the chemical sensing experiment in this thesis from most others is the use of ambient air and a long measurement period, as opposed to short experiments performed in a controlled environment. A long-term experiment in ambient air will better indicate the sensor's actual performance. Although researchers in [95] found that a CMUT sensor was proved to detect CO<sub>2</sub> and estimate humidity, and Stedman [9], used the CMUT sensor to classify five chemicals and to separate coffee beans by their odour. It is important to note that the experiments were all conducted in controlled environments. The chemical sensing experiment conducted in the greenhouse will better show the behaviour and drift of the functional layers.

### 3.3 Comparing the CMUT sensor to end-user requirements

To evaluate the CMUT sensors performance, the CMUT sensor is compared to standards set by DARPA. Their standards are: a chemical sensor should be able to recognize at least 5 different chemical warfare agents, determine each chemical with a probability of 85% and the probability for a false alarm should be 10<sup>-5</sup> %.

An ability of recognizing different chemicals has already been shown by Stedman in [9], where the sensor distinguished 5 differed chemical compounds with a 90% probability. Despite not sensing warfare agents, an excellent selectivity is shown. Reaching the first goal set by DARPA.

One might be concerned about the detection limit of the sensor as warfare agents and explosives have low concentrations. For example; the short time exposure limit to mustard gas, recognized by its DMMP concentration, is 17.4 ppb . In [96] researchers proved that the CMUT sensor could be used to detect concentration down to 10 ppb of DMMP. Proving the sensor to also be sensitive, reaching another requirement set by DARPA.

No previous research, known to the author, has provided a probability of false alarm. This thesis showed the best probability for false positives to be 10<sup>-1</sup>. Compared to the 10<sup>-5</sup> required by DARPA the sensor used in the thesis will not be approved. With this in mind, it is important to note that the machine learning models' prediction can be better if the dataset has a larger

variety with more samples. As a preparation for the thesis the data Stedman provided in his Ph.D. thesis [9], was used for preparation. When testing for false positives a probability of  $10^{-5}$  was found for separating all the five chemicals from each other. The data Stedman collected was short time, so no drift could have taken place, helping explain the deviation between the numbers. Emphasis should be placed on collecting more data as a consequence of the findings.

### 3.4 Comparison with available chemical sensor solutions

Compared to other available chemical sensors solutions, the CMUT sensor is a strong contender. Even though little information is found about the available sensors, due to competition, some common features are found. They are small, expensive, selective and have a short lifetime. Compared to the chemical CMUT sensor which is large, selective, low cost and has a short lifetime. The CMUT sensor is promising due to its price, and issues with size are already being handled. A suggested way to enhance the sensors lifetime would be to develop better polymers or finding a way to handle the sensor drift as mentioned earlier.

A trend in processing data from the sensor in the “cloud” is also seen amongst the available sensors, especially the products aiming at larger markets. For example, CHEMISENSE, presented in 1.3.4, aiming at the consumer market by making a larger heat map, integrates a cloud solution in their product. This enables scalability, faster data processing, recalibration and helps make larger sensor networks possible. The ability of tuning and updating the sensor software over the cloud can also be achieved. Some of the sensors in the review also have a poor selectivity, but, as discussed earlier, the chemical CMUT sensor has excellent sensitivity although drift is observed. Although many sensors are on the market no large commercial success is seen. One might speculate that this is due to the necessity of a large data library, allowing to look up fingerprints for different chemicals and relate them to known concentrations. Developing such a library is a time consuming and expensive process, so methods for reducing time and cost should be considered. An example of such a method is implementing an autosampler.

### 3.5 Autosampler allowing efficient sampling of data

The autosampler proved itself as an efficient and reliable sampling machine by showing capabilities of sampling continuously for 30 hours without problems. The design of the autosampler allows for easy modification both in the hardware and software if changes to the test setup is made. Although successful, the scalability of the robot is limited to 180 samples due to its working area. If more samples are needed another robot must be acquired. Restrictions of the autosampler is that there is no way of securely knowing that it does not puncture the vial at the same spot twice. Nor does it have any way of giving feedback, for example if the arm has gone off track or that the samples are missed. To avoid mistakes like these a camera can be mounted on the frame for remote monitoring.



Other automation processes identified are the data preprocessing for the gas chromatography and watering and irrigation of plants. In the continuation of the project a Fourier transform infrared spectroscopy (FTIR) sampling machine will substitute the GC to increase the sampling rate. As a result, a larger dataset will be collected for future testing which will give a more precise picture of the capabilities of the sensor.





## 4 CONCLUSION

The aim of this thesis was to determine a proof of concept of a CMUT chemical sensor technology as a gas detecting unit that can classify and quantify chemicals with machine learning in a real-world situation. Overall the CMUT sensor could detect CO<sub>2</sub> in ambient air down to 120 ppm, although suffering from data drift and unbalanced data the machine learning model could classify and quantify CO<sub>2</sub>. Data drift due to polymer degradation needs to be further researched, more data should be collected over longer time periods.

As a commercial product, the CMUT sensor can be viable if the sensor drift and miniaturization is fixed, compared to DARPA's criteria the sensor did not pass based on the results in the thesis. Although previous research has shown promising capabilities. More testing in realistic environments should be done. Also sampling data and training machine learning models in the cloud should be implemented to be able to compete with electronic noses on market today.

The autosampler developed proved itself successful during testing and will contribute to make future data collection effective. An important contribution to the development of the machine learning database. And hopefully an important contribution for further testing of the CMUT chemical sensor.

Detecting compounds in the air has been a time consuming and expensive process. By utilizing the autosampler and modular machine learning models testing novel sensor technology can be improved. And the CMUT chemical sensor could be the inexpensive, robust and portable sensor we have been waiting for.



## 5 SUGGESTED FUTURE WORK

To further improve the CMUT chemical sensor and the testing procedure following steps are presented

### **Focus on eliminating drift in dataset**

Drift in the dataset contributed to poor machine learning models further research on how to eliminate the dataset drift should be conducted. Also suggested in the discussion is to eliminate drift through software by using orthogonal signal correction, fast Fourier transform or discrete wavelet transform for correcting drift [93]. Another interesting way to correct dataset drift, is through the use of deep learning. In [97] researchers propose a drift compensation for a gas sensor using a deep learning model, showing it to be successful and outperforming the regular SVM model. This should be looked further into. Deep learning models are not useful on small datasets as there is a great risk of overfitting, so the deep learning method should be tested on a larger dataset than the one gathered for the CMUT sensor. Consequently, more data must be gathered.

### **Sample more data and develop standard for database**

Machine learning models are only as good as the data the model is based on. In the case of machine learning, more data opens the possibilities to more algorithms as well as ensuring that the prediction is based on a good foundation, eliminating the risk of overfitting and allowing generalization of the data. Arguably one can not put more emphasis on how important it is to collect a high quality dataset. With high quality meaning a large amount of evenly distributed data. The importance for long time testing in ambient air is also important as it reveals how the product will behave in real-world situations. Researchers should think about a standard for collecting data. So that the collected from early stage testing can be useful at later stages and used in the machine learning database. Having a standard collecting procedure, and data format also allows the machine learning models to be reused, reducing work.

### **Implement the autosampler as a part of the field test**

The autosampler developed should be integrated in the field testing setup. Timers need to be set in the Arduino and autosampler code so that the autosampler moves in correct order in accordance to the sampling script already developed. A better way of implementing the autosampler system should also be researched. For example, replacing the Arduino chip with an IoT chip to allow remote controlling of the circuit is one example of improvement. Further the pump and valve circuit can be miniaturized by making a circuit board from. The valves also need replacing as they proved not to be thrust worthy. More vial boards and a better way to secure them should also be done before implementing the autosampler in the hardware set up by Stanford.



**Miniaturization of the sensor and cloud connection**

Reducing the sensor size increases the application areas as well as increasing the competitiveness of the sensor. Miniaturization of the sensor enables it to be placed and carried around almost anywhere. As seen in available chemicals sensor solutions, available products connecting the sensor to the cloud is an important step towards a larger networks of sensors. Cloud connection could be done by implementing a WiFi or a cellular WiFi on the chip and regularly uploading the data collected by the sensor. Several companies such as Amazon, IBM and Azure deliver database solutions and the possibility for implementing machine learning models.



## 6 BIBLIOGRAPHY

1. Dragonieri, S., et al., *Electronic nose technology in respiratory diseases*. Lung, 2017. **195**(2): p. 157-165.
2. Zampolli, S., et al., *An electronic nose based on solid state sensor arrays for low-cost indoor air quality monitoring applications*. Sensors and Actuators B: Chemical, 2004. **101**(1-2): p. 39-46.
3. Sayago, I., et al., *Graphene oxide as sensitive layer in Love-wave surface acoustic wave sensors for the detection of chemical warfare agent simulants*. Talanta, 2016. **148**: p. 393-400.
4. World Health Organization. Air pollution 2018 06.04.18]; Available from: <http://www.who.int/airpollution/en/>.
5. World Health Organization. *WHO concerned about suspected chemical attacks in Syria*. 2018 [cited 2018 14.04]; Available from: <http://www.who.int/mediacentre/news/statements/2018/chemical-attacks-syria/en/>.
6. Gardner, J.W. and P.N. Bartlett, *A brief history of electronic noses*. Sensors and Actuators B: Chemical, 1994. **18**(1-3): p. 210-211.
7. Wilson, A.D., *Review of electronic-nose technologies and algorithms to detect hazardous chemicals in the environment*. Procedia Technology, 2012. **1**: p. 453-463.
8. Laboratory, E.L.G. *Stanford: Khuri-Yakib Group*. 2018 [cited 2018 01.04]; Available from: <https://kyg.stanford.edu/>.
9. Stedman, G.Q., *Chemical Sensing with Capacitive Micromachined Ultrasonic Transducers*. 2017, Stanford University.
10. The Engineering Toolbox. *Air - Composition and Molecular Weight*. 2018 [cited 2018 16.03]; Available from: [https://www.engineeringtoolbox.com/air-composition-d\\_212.html](https://www.engineeringtoolbox.com/air-composition-d_212.html).
11. Compagnone, D., et al., *Chemical Sensors and Biosensors in Italy: A Review of the 2015 Literature*. Sensors, 2017. **17**(4): p. 868.
12. Chang, Y.-P. and Y.-H. Chu, *Gas Sensing Ionic Liquids on Quartz Crystal Microbalance, in Progress and Developments in Ionic Liquids*. 2017, InTech.
13. Wen, W., *Introductory Chapter: What is Chemical Sensor?*, in *Progresses in Chemical Sensor*. 2016, InTech.
14. Mujahid, A. and F.L. Dickert, *Surface Acoustic Wave (SAW) for Chemical Sensing Applications of Recognition Layers*. Sensors, 2017. **17**(12): p. 2716.
15. Jones, R.G. and U.i.d.c.p.e.a.P. Division, *Compendium of Polymer Terminology and Nomenclature: IUPAC Recommendations, 2008*. 2009: Royal Society of Chemistry Cambridge.
16. The Editors of Encyclopædia Britannica. *Polymer*. 2017 [cited 2018 02.02]; Available from: <https://www.britannica.com/science/polymer>.
17. Tyagi, S. and V.G. Mahesh, *Saw and interdigital transducers*. International Journal of Scientific & Engineering Research, 2012. **3**(12): p. 1-4.
18. Barié, N., et al., *Detection of coffee flavour ageing by solid-phase microextraction/surface acoustic wave sensor array technique (SPME/SAW)*. Food chemistry, 2015. **176**: p. 212-218.
19. Hribšek, M.F., et al., *Modelling of chemical surface acoustic wave sensors and comparative analysis of new sensing materials*. International Journal of Numerical Modelling: Electronic Networks, Devices and Fields, 2013. **26**(3): p. 263-274.
20. Hamidon, M.N. and Z. Yunusa, *Sensing Materials for Surface Acoustic Wave Chemical Sensors, in Progresses in Chemical Sensor*. 2016, InTech.
21. Devkota, J., P.R. Ohodnicki, and D.W. Greve, *SAW sensors for chemical vapors and gases*. Sensors, 2017. **17**(4): p. 801.
22. Pearce, T.C., et al., *Handbook of machine olfaction: electronic nose technology*. 2006: John Wiley & Sons.
23. Arshak, K., et al., *A review of gas sensors employed in electronic nose applications*. Sensor review, 2004. **24**(2): p. 181-198.
24. Initium. *Technology, piezoelectric effect*. 2009 [cited 2018 06.02]; Available from: <http://www.initium2000.com/en/technology.html>.
25. Vaughan, R., C. O'sullivan, and G. Guilbault, *Development of a quartz crystal microbalance (QCM) immunosensor for the detection of Listeria monocytogenes*. Enzyme and Microbial Technology, 2001. **29**(10): p. 635-638.



26. Arnau, A., T. Sogorb, and Y. Jimenez, *QCM100-quartz crystal microbalance theory and calibration*. Rev. Sci. Instrum, 2000. **71**: p. 2563.
27. Caruso, F., et al., *Quartz crystal microbalance and surface plasmon resonance study of surfactant adsorption onto gold and chromium oxide surfaces*. Langmuir, 1995. **11**(5): p. 1546-1552.
28. Hoffmann, R., M. Schreiter, and J. Heitmann, *The concept of thin film bulk acoustic resonators as selective CO<sub>2</sub> gas sensors*. Journal of Sensors and Sensor Systems, 2017. **6**(1): p. 87.
29. Datta, S. *Intro to Piezoelectric BAW Resonator Modeling*. 2013 [cited 2018 10.02]; Available from: <https://www.comsol.com/blogs/piezoelectric-baw-resonator-modeling/>.
30. Ruby, R.C., et al. *Thin film bulk wave acoustic resonators (FBAR) for wireless applications*. in *Ultrasonics Symposium, 2001 IEEE*. 2001. IEEE.
31. Johnston, M.L., I. Kymissis, and K.L. Shepard, *FBAR-CMOS oscillator array for mass-sensing applications*. IEEE Sensors Journal, 2010. **10**(6): p. 1042-1047.
32. Vashist, S.K., *A review of microcantilevers for sensing applications*. Journal of Nanotechnology Online, 2007. **3**(11): p. 1113-1128.
33. Sepaniak, M., et al., *Peer reviewed: Microcantilever transducers: A new approach in sensor technology*. 2002, ACS Publications.
34. del Rey, M., et al., *Monitoring swelling and deswelling of thin polymer films by microcantilever sensors*. Sensors and Actuators B: Chemical, 2014. **204**: p. 602-610.
35. Baller, M., et al., *A cantilever array-based artificial nose*. Ultramicroscopy, 2000. **82**(1-4): p. 1-9.
36. Fraunhofer Institute for Photonic Microsystems. *Micromachined Ultrasonic Transponders in Post-CMOS Technology*. 2018 [cited 2018 03.04]; Available from: <https://www.ipms.fraunhofer.de/en/research-development/cmud.html>.
37. Ergun, A.S., G.G. Yaralioglu, and B.T. Khuri-Yakub, *Capacitive micromachined ultrasonic transducers: Theory and technology*. Journal of aerospace engineering, 2003. **16**(2): p. 76-84.
38. Salim, M.S., et al., *Capacitive micromachined ultrasonic transducers: technology and application*. Journal of Medical Ultrasound, 2012. **20**(1): p. 8-31.
39. Ramanaviciene, A., et al., *Capacitive micromachined ultrasound transducer (cMUT) for immunosensor design*. Analyst, 2010. **135**(7): p. 1531-1534.
40. Barauskas, D., et al., *Greenhouse gas molecule CO<sub>2</sub> detection using a capacitive micromachined ultrasound transducer*. Analytical chemistry, 2016. **88**(13): p. 6662-6665.
41. Stedman, Q., K.K. Park, and B.T. Khuri-Yakub. *cMUT chip with integrated temperature and pressure sensors*. in *Ultrasonics Symposium (IUS), 2016 IEEE International*. 2016. IEEE.
42. Shaik, M., et al., *p-Hexafluoroisopropanol phenyl functionalized graphene for QCM based detection of dimethyl methylphosphonate, a simulant of the nerve agent sarin*. RSC Advances, 2018. **8**(15): p. 8240-8245.
43. Chen, D., et al., *Nerve gas sensor using film bulk acoustic resonator modified with a self-assembled Cu<sub>2</sub>+/11-mercaptopundecanoic acid bilayer*. Sensors and Actuators B: Chemical, 2010. **150**(1): p. 483-486.
44. Cannatà, D., et al., *Nerve agent simulant detection by solidly mounted resonators (SMRs) polymer coated using laser induced forward transfer (LIFT) technique*. Sensors and Actuators B: Chemical, 2012. **173**: p. 32-39.
45. Lee, H.J., et al., *Chemical vapor detection using a capacitive micromachined ultrasonic transducer*. Analytical chemistry, 2011. **83**(24): p. 9314-9320.
46. Flewitt, A. *FBAR Sensor - The Portable Mass Sensor for Gas and Biological Sensing*. [cited 2018 14.02]; Available from: <https://www.enterprise.cam.ac.uk/wp-content/uploads/2012/10/Marketing-Sheet-Biosensing-FBAR.pdf>.
47. Nature. *Metal-organic frameworks*. 2018 [cited 2018 14.02]; Available from: <https://www.nature.com/subjects/metal-organic-frameworks>.
48. Matrix sensors. *Gas sensors for a cleaner, healthier, safer world*. 2018 [cited 2018 30.01]; Available from: <http://matrixsensorsinc.com/>.
49. Boehle, T. *large pore of MOF-5 with yellow ball*. [Figure] 2010 [cited 2018 15.02]; Available from: [https://commons.wikimedia.org/wiki/File:IRMOF-1\\_wiki.png](https://commons.wikimedia.org/wiki/File:IRMOF-1_wiki.png).
50. CNSI UCLA. *Matrix sensors*. 2018 [cited 2018 15.02]; Available from: <https://cnsi.ucla.edu/project/matrix-sensors/>.



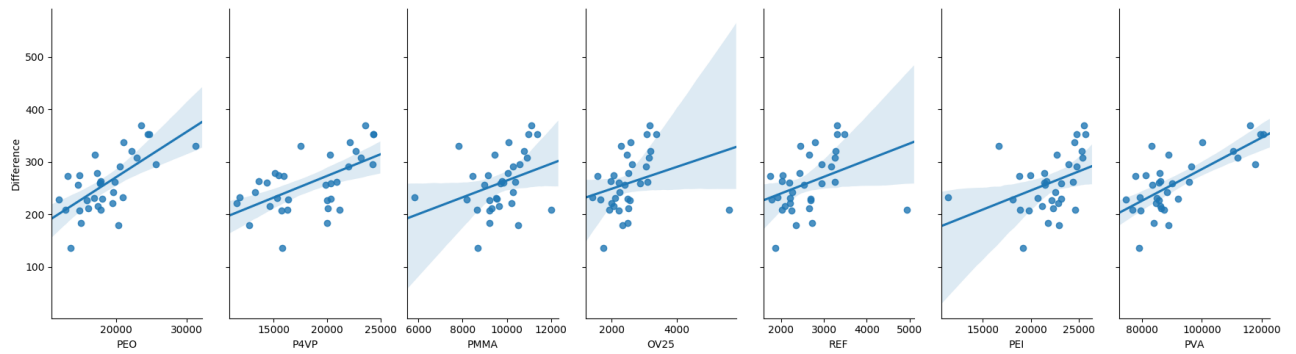
51. OPEN QCM. *Technology*. 2014 [cited 2018 03.02]; Available from: <http://openqcm.com/technology>.
52. Open QCM. *Gallery*. 2015 [cited 2018 03.02]; Available from: <http://openqcm.com/gallery/openqcm-pictures>.
53. Hraybi, A., et al. *Quartz crystal microbalance (QCM) for the detection of explosive vapor—measurements and simulations*. in *Landmine: Detection, Clearance and Legislations (LDCL), 2017 First International Conference on*. 2017. IEEE.
54. Della Ventura, B., et al., *Effective antibodies immobilization and functionalized nanoparticles in a quartz-crystal microbalance-based immunosensor for the detection of parathion*. PloS one, 2017. **12**(2): p. e0171754.
55. Open QCM. *Store openQCM*. 2018 [cited 2018 03.02]; Available from: <https://store.openqcm.com/>.
56. angel.co. *ChemiSense*, . 2014 [cited 2018 05.02]; Available from: <https://angel.co/chemisense>.
57. ChemiSense. *Advanced air quality monitoring* 2015 [cited 2018 06.02]; Available from: <https://chemisense.co/>.
58. Metz, R. *An Air-Quality Monitor You Take with You*. 2014 [cited 2018 18.02]; Available from: <https://www.technologyreview.com/s/530011/an-air-quality-monitor-you-take-with-you/>.
59. Nore, P.W., *Pollution Detection in a Low-Cost Electronic Nose, a Machine Learning Approach*. 2016.
60. common invent. *enose - Online air quality monitoring technology*. 2018 [cited 2018 07.05]; Available from: <http://www.comon-invent.com/enose/>.
61. Copeland, M. *What's the Difference Between Artificial Intelligence, Machine Learning, and Deep Learning?* 2016 [cited 2018 18.02]; Available from: <https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/>.
62. Srivastava, T. *Difference between Machine Learning & Statistical Modeling*. 2015 [cited 2018 26.04]; Available from: <https://www.analyticsvidhya.com/blog/2015/07/difference-machine-learning-statistical-modeling/>.
63. Rodriguez Lujan, I., J. Fonollosa Magrinyà, and R. Huerta. *Machine learning methods in electronic nose analysis*. in *book of proceedings*. 2016.
64. Udacity. *Machine learning engineer nanodegree program; Supervised Learning Intro*. 2018 [cited 2018 22.03]; Available from: <https://classroom.udacity.com/nanodegrees/nd009/parts/6fbc8ac7-7d1e-4cb1-8940-02af4efb51bc>.
65. Danilo Bzdok, M.K., Naomi Altman. *Points of significance: Machine learning: supervised methods*. 2018 [cited 2018 26.04]; Available from: <https://www.nature.com/articles/nmeth.4551>.
66. Müller, A.C. and S. Guido, *Introduction to machine learning with Python: a guide for data scientists*. 2016: " O'Reilly Media, Inc."
67. Chernozhukov, V., C. Hansen, and Y. Liao, *A lava attack on the recovery of sums of dense and sparse signals*. *The Annals of Statistics*, 2017. **45**(1): p. 39-76.
68. Andrew Ng, J.N., Chuan Yu Foo, Yifan Mai, Caroline Suen, Adam Coates, Andrew Maas, Awni Hannun, Brody Huval, Tao Wang, Sameep Tandon. *Logistic regression, UFLDL tutorial*. 2018 [cited 2018 27.04]; Available from: <http://ufldl.stanford.edu/tutorial/supervised/LogisticRegression/>.
69. Simonoff, J., *Logistic regression—modeling the probability of success*. Teaching paper of Leonard N. Stern School of Business, New York University, 2012.
70. Cambridge Spark. *Support Vector Machines*. 2018 [cited 2018 07.05]; Available from: <http://beta.cambridgespark.com/courses/jpm/05-module.html>.
71. Bell, J., *Machine learning: hands-on for developers and technical professionals*. 2014: John Wiley & Sons.
72. Samarasinghe, S., *Neural networks for applied sciences and engineering: from fundamentals to complex pattern recognition*. 2016: CRC Press.
73. scikit learn. *Cross-validation: evaluating estimator performance*. 2018 [cited 2018 10.05]; Available from: [http://scikit-learn.org/stable/modules/cross\\_validation.html](http://scikit-learn.org/stable/modules/cross_validation.html).



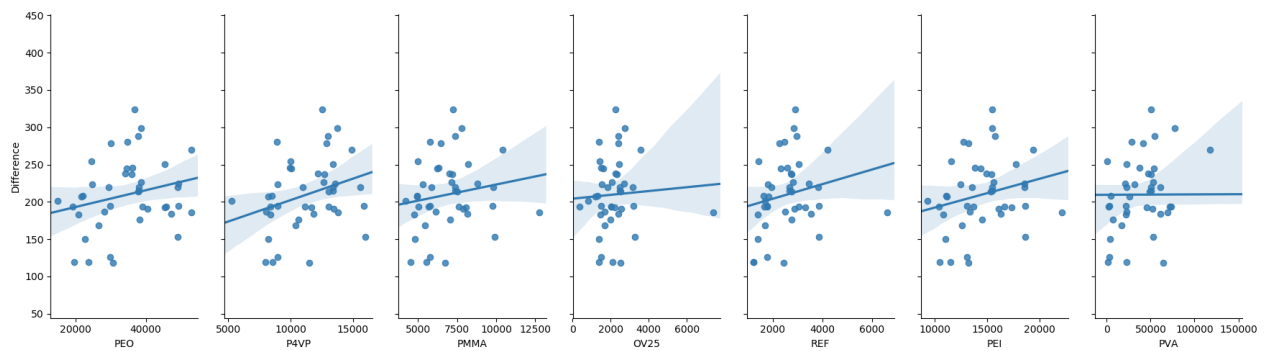
74. Zheng, A., *Evaluating Machine Learning Models*. Dostopano: <https://www.oreilly.com/ideas/evaluating-machinelearning-models/page/5/hyperparameter-tuning>. [Dostopano: 20. 8. 2017], 2015. **40**: p. 15.
75. David, Z. *Linear regression 101 (Part 2 - Metrics)*. 2018 [cited 2018 09.05]; Available from: <https://dziganto.github.io/data-science/linear-regression/machine-learning/python/Linear-Regression-101-Metrics/>.
76. Stanford University. *Lecture 11: Corss validation STATS 202: Data mining and analysis*. 2017 [cited 2018 09.05]; Available from: <https://web.stanford.edu/class/stats202/content/lec11-cond.pdf>.
77. Defense Advanced Research Projects Agency, *Board Agency Announcent SIGMA+ Sensors*, D.S. Office, Editor. 2018.
78. Omberg, K., *Greenhouse project*, M. Byrne, Editor. 2018.
79. opentrons. *Our Robots*. 2017 [cited 2018 02.02]; Available from: <http://opentrons.com/robots>.
80. Enoch, H. and S. Dasberg, *The occurrence of high CO<sub>2</sub> concentrations in soil air*. *Geoderma*, 1971. **6**(1): p. 17-21.
81. Kyaw Thet, N.W. *Gas Chromatography*. 2015 [cited 2018 04.02]; Available from: [https://chem.libretexts.org/Core/Analytical\\_Chemistry/Instrumental\\_Analysis/Chromatography/Gas\\_Chromatography](https://chem.libretexts.org/Core/Analytical_Chemistry/Instrumental_Analysis/Chromatography/Gas_Chromatography).
82. Sheffield Hallam University. *Gas chromatography* 2017 [cited 2018 06.02]; Available from: <https://teaching.shu.ac.uk/hwb/chemistry/tutorials/chrom/gaschrm.htm>.
83. Welsh, R., *Gas Chromatograph*. 2017.
84. scikit learn. *scikit-learn, Machine Learning in Python*. 2018 [cited 2018 10.05]; Available from: <http://scikit-learn.org/stable/>.
85. NumPy. *Numpy*. 2018 [cited 2018 10.05]; Available from: <http://www.numpy.org/>.
86. Pandas. *Python Data Analysis Library*. 2018 [cited 2018 10.05]; Available from: <https://pandas.pydata.org/>.
87. matplotlib. *matplotlib version 2.2.2*. 2018 [cited 2018 10.05]; Available from: <https://matplotlib.org/>.
88. Scikit learn. *sklearn.model\_selection.train\_test\_split*. 2018 [cited 2018 03.05]; Available from: [http://scikit-learn.org/stable/modules/generated/sklearn.model\\_selection.train\\_test\\_split.html](http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html).
89. Scikit learn. *sklearn.preprocessing.StandardScaler*. 2018 [cited 2018 20.04]; Available from: <http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html>.
90. Scikit learn. *sklearn.svm.SVC*. 2018 [cited 2018 20.04]; Available from: <http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html>.
91. scikit learn. *sklearn.model\_selection.GridSeachCV*. 2018 [cited 2018 22.04]; Available from: [http://scikit-learn.org/stable/modules/generated/sklearn.model\\_selection.GridSearchCV.html](http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html).
92. scikit learn. *RBF SVM parameters*. 2018 [cited 2018 20.04]; Available from: [http://scikit-learn.org/stable/auto\\_examples/svm/plot\\_rbf\\_parameters.html](http://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html).
93. Di Carlo, S. and M. Falasconi, *Drift correction methods for gas chemical sensors in artificial olfaction systems: techniques and challenges*, in *Advances in Chemical Sensors*. 2012, InTech.
94. Netherlands Technical Agreement, *Air quality - Electronic air monitoring - Odour and safety (NTA 9055)*. 2012.
95. Lee, H.J., et al., *Mesoporous thin-film on highly-sensitive resonant chemical sensor for relative humidity and CO<sub>2</sub> detection*. *Analytical chemistry*, 2012. **84**(7): p. 3063-3066.
96. Lee, H.J., et al. *CMUT as a chemical sensor for DMMP detection*. in *Frequency Control Symposium, 2008 IEEE International*. 2008. IEEE.
97. Liu, Q., et al., *Gas recognition under sensor drift by using deep learning*. *International Journal of Intelligent Systems*, 2015. **30**(8): p. 907-922.

# APPENDIX

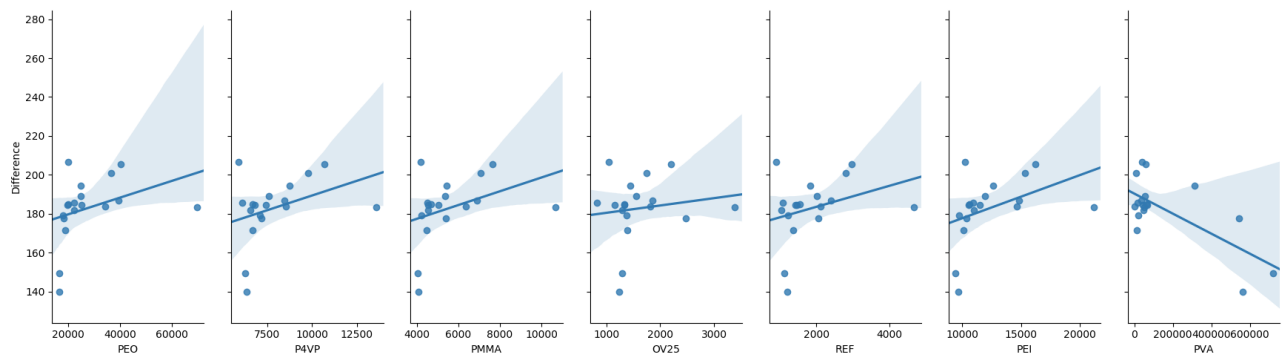
## Appendix A: Linear regression models on a weekly basis



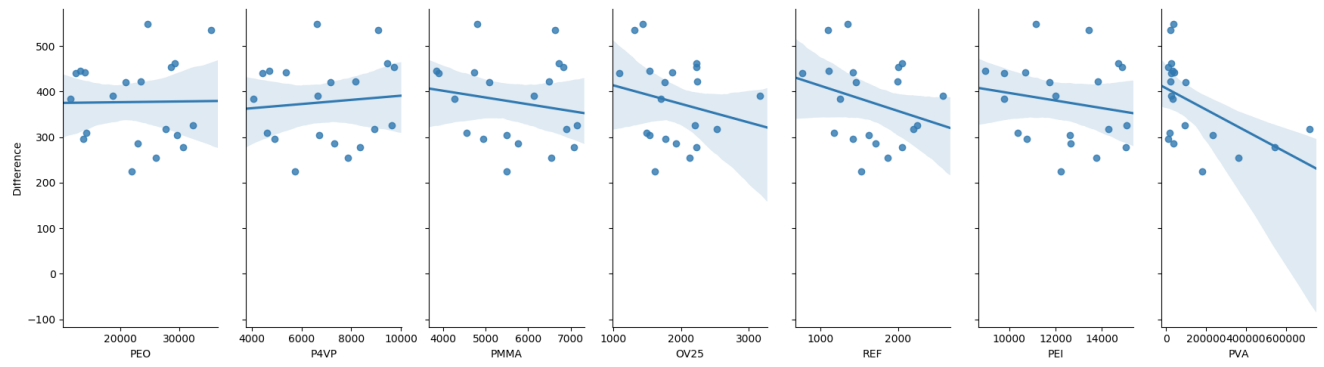
**Figure 47: Scatter plot and the regression line with 95% confidence interval of the CO<sub>2</sub> difference and the different frequency responses for week 44 with the gas measuring tube**



**Figure 48: Scatter plot and the regression line with 95% confidence interval of the CO<sub>2</sub> difference and the different frequency responses for week 44 with bubble column**



**Figure 49: Scatter plot and the regression line with 95% confidence interval of the CO<sub>2</sub> difference and the different frequency responses for week 44-45 with the gas measuring tube**



**Figure 50: Scatter plot and the regression line with 95% confidence interval of the CO<sub>2</sub> difference and the different frequency responses for week 44-45 with the gas measuring tube**



## Appendix B: Code flow charts

### B.1 Jupyter notebook code

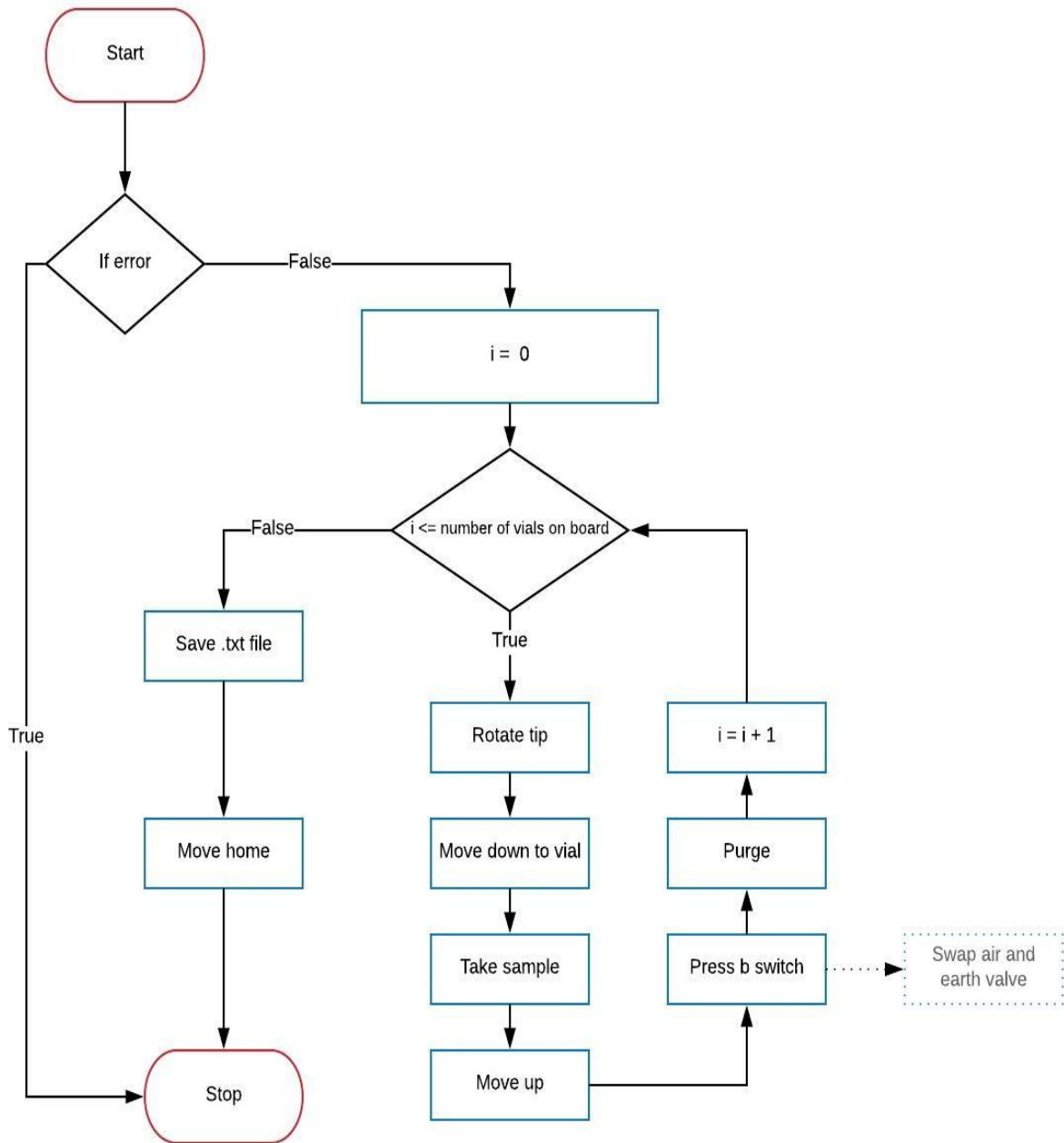


Figure 51: Flow diagram for the main process of the robot arm

## B.2 Valve and pump circuit code

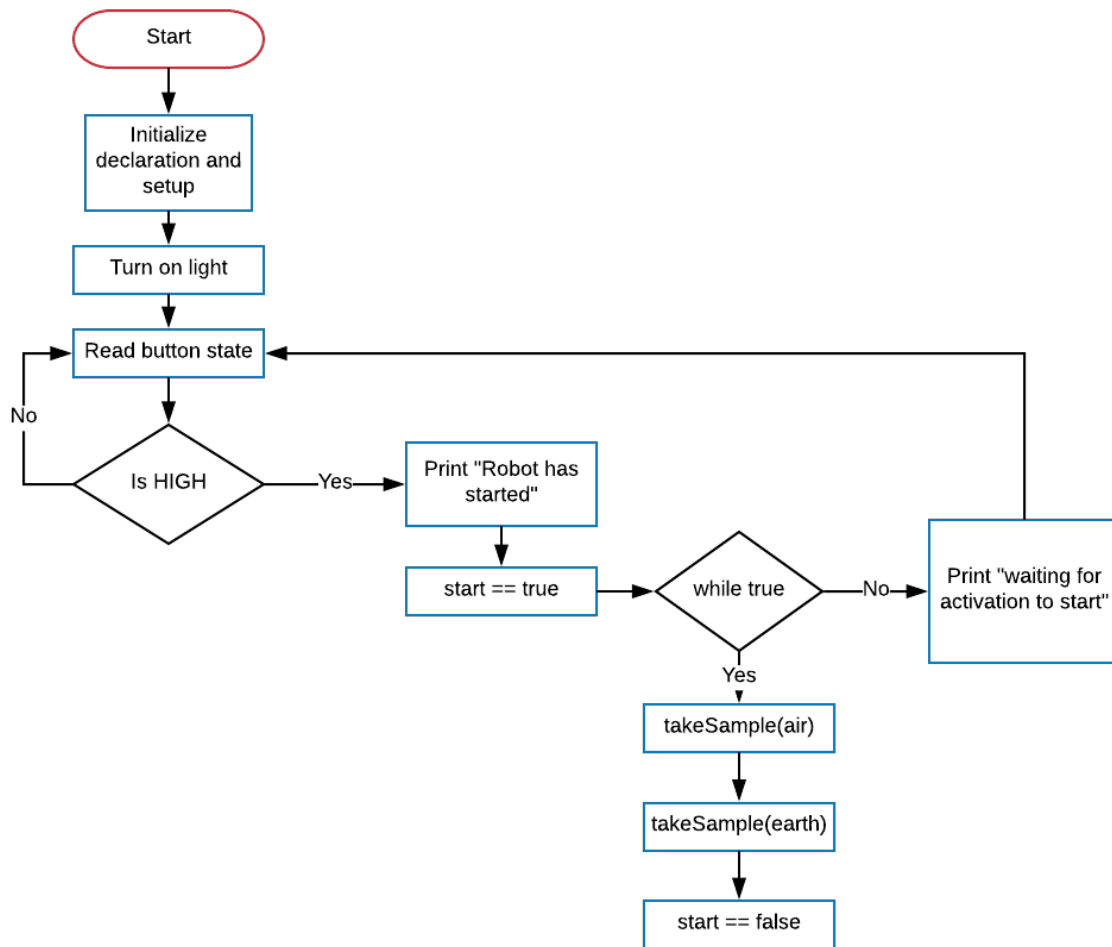


Figure 52: Flow chart of the Arduino code for the valve circuit

### B.3 TakeSample function

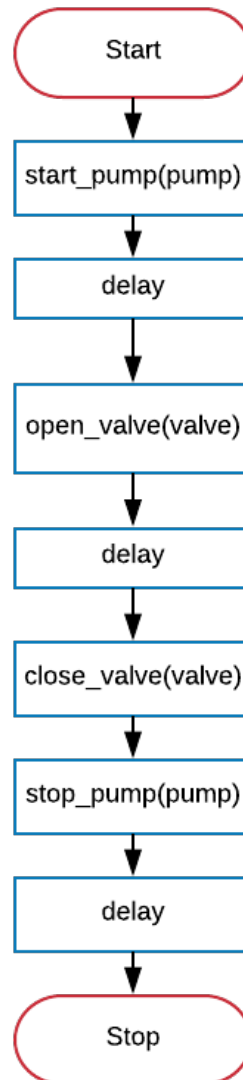


Figure 53: Take sample function



## Appendix C: Source code and data

### C.1 Source code

To download original files go to:

[https://eduumb-my.sharepoint.com/:f/g/personal/krom\\_nmbu\\_no/EoeRjNSSmSIEkbOAPAsYG1wB0oavZ5FAw1oV2Bc\\_exuoQA?e=exoejJ](https://eduumb-my.sharepoint.com/:f/g/personal/krom_nmbu_no/EoeRjNSSmSIEkbOAPAsYG1wB0oavZ5FAw1oV2Bc_exuoQA?e=exoejJ)

All scripts are presented below:

### CLASSIFICATION.py

Script for performing classification in python, returns Crossvalidation and accuracy scores as well as confusion matrixes.

```

1. import numpy as np
2. import matplotlib.pyplot as plt
3. import pandas as pd
4. import itertools
5. from time import time
6. from collections import Counter
7. from sklearn import grid_search
8. from sklearn.cross_validation import train_test_split, LeaveOneOut
9. from sklearn.preprocessing import StandardScaler
10. from sklearn.linear_model import LogisticRegression, Ridge
11. from sklearn.pipeline import make_pipeline
12. from sklearn.svm import SVC
13. from sklearn.metrics import confusion_matrix
14. from sklearn.model_selection import GridSearchCV
15. from sklearn.metrics import classification_report
16. from sklearn.pipeline import Pipeline
17. from sklearn.neural_network import MLPClassifier
18.
19. #Import data file
20. excel_file = 'co2_measurements.xlsx'
21.
22. held_out1_ = [[-37903, -13031, -7362, -2427, -2955, -15761, -54334]]
23. held_out2_ = [[-21094, -22111, -10072, -2591, -2801, -24575, -100092]]
24. print(held_out1_)
25. #Convert excel fie to panda dataframe, skip the two first rows as they do not conta
    in usefull information
26. measurements_raw = pd.read_excel(excel_file, "Sensor_Responses", skiprows=2)
27. #Drop columns with autogenerated data
28. measurements_drop = measurements_raw.drop(columns=['Unnamed: 0', 'Earth.1', 'Air.1',
29.                                                    'Difference.1', 'Unnamed: 16',
30.                                                    'OK?', 'Sample Number'])
31. #Drop rows with more than 2 missing values
32. measurements = measurements_drop.dropna(thresh = 2)
33. #Standard scaler object
34. sc = StandardScaler()
35.
36. #Set frequency responses as features
37. features_ = measurements.loc[:, 'PEO':'PVA']
38. print(len(features_))
39. #Set target values, for concentratitons over 350ppm = 1, lower than 350 = 0
40. target_df = pd.DataFrame(np.where(measurements['Difference'] >= 350, 1, 0))

```



```

41. print(target_df.count())
42. #convert to array
43. target = target_df.as_matrix().ravel()
44. #print size of target array
45. print(Counter(target))
46. #scale features and the two ledt out datapoints
47. features = sc.fit_transform(features_)
48. held_out1 = sc.transform(held_out1_)
49. held_out2 = sc.transform(held_out2_)
50. def plot_confusion_matrix(cm, classes,
51.                             normalize=False,
52.                             title='Confusion matrix',
53.                             cmap=plt.cm.Blues):
54.     """
55.     This function prints and plots the confusion matrix.
56.     Normalization can be applied by setting `normalize=True`.
57.     """
58.     if normalize:
59.         cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
60.         print("Normalized confusion matrix")
61.     else:
62.         print('Confusion matrix, without normalization')
63.
64.     print(cm)
65.     plt.imshow(cm, interpolation='nearest', cmap=cmap)
66.     plt.title(title)
67.     plt.colorbar()
68.     tick_marks = np.arange(len(classes))
69.     plt.xticks(tick_marks, classes, rotation=45)
70.     plt.yticks(tick_marks, classes)
71.
72.     fmt = '.2f' if normalize else 'd'
73.     thresh = cm.max() / 2.
74.     for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
75.         plt.text(j, i, format(cm[i, j], fmt),
76.                  horizontalalignment="center",
77.                  color="white" if cm[i, j] > thresh else "black")
78.
79.     plt.tight_layout()
80.     plt.ylabel('True label')
81.     plt.xlabel('Predicted label')
82.     return
83.
84. #Function that selects the best parameters for the SVM model with three different k
    ernels
85. def svc_param_selection(X, y, nfolds):
86.     """
87.     inputs:
88.     X - feature data
89.     y - target data
90.     nfolds- amout of folds if grid search with corsss validation
91.     is to be performed
92.     """
93.     Cs = [0.01, 0.1, 1, 10, 100]
94.     gammas = [0.01, 0.1, 1, 10, 100]
95.     kernels = ['poly', 'rbf', 'sigmoid']
96.     param_grid = {'C': Cs, 'gamma' : gammas, 'kernel':kernels}
97.     #grid_search = GridSearchCV(svm.SVC(kernel='poly'), param_grid, cv=nfolds)
98.     grid_search = GridSearchCV(SVC(), param_grid)
99.     #fit model to data
100.     loo = LeaveOneOut(nfolds)
101.     grid_search.fit(X, y)
102.     #Find best parameters based on accuracy
103.     grid_search.best_params_
104.     return grid_search.best_params_
105.

```



```

106.     print(svc_param_selection(features, target, 20))
107.     #Function to make prediction though pipeline
108.     def ml_pipeline(data_x, data_y, learner):
109.         """
110.             Function returns cross validation score of the machine learning mode
111.         """
112.         inputs:
113.             data_x - feature data
114.             data_y - target data
115.             learner- machine learning models
116.         #Make pipeline with standard scaler and classifier of choice
117.         pipeline = make_pipeline(sc, learner)
118.         #Split data
119.         X_train, X_test, y_train, y_test = train_test_split(data_x, data_y, test
120. _size=0.40, random_state=42)
121.         #Start timer for finding prediction time
122.         start = time()
123.         #fit pipeline on data
124.         pipeline.fit(X_train, y_train)
125.         #Predict on X_test data to make test predictions
126.         y_pred = pipeline.predict(X_test)
127.         #Stop timer
128.         stop = time()
129.         #Make classification report,gives more info than the cross val score.
130.         report = classification_report(y_test, y_pred)
131.         #Calculate prediction time
132.         pred_time = stop-start
133.         print("Prediction time", pred_time)
134.         print("Predicted by learner", learner.predict(held_out2))
135.         #Plot confusion matrix with normalization
136.         cnf_matrix = confusion_matrix(y_test, y_pred)
137.         np.set_printoptions(precision=2)
138.         plt.figure()
139.         plot_confusion_matrix(cnf_matrix, classes = ["HIGH", "LOW"], normalize =
140. True, title='
141. ')
142.         return report
143.
144.     #Initializing machine learning models
145.     #CHANGE C and gamma parameters for tuning
146.     log_reg = LogisticRegression(C=20)
147.     SVM_linear = SVC(kernel='sigmoid', gamma=1, C=10)
148.     SVM_gauss = SVC(kernel='rbf', gamma=0.5, C=100)
149.     SVM_poly = SVC(kernel='poly', gamma=0.2, C=1)
150.     mlp = MLPClassifier(hidden_layer_sizes=(2,2), activation = 'relu', solver =
151. 'lbfgs')
152.
153.     #Loop though list of the differnt machine learning models and run them in th
154. e pipeline
155.     for clf in [log_reg, SVM_gauss, SVM_linear, SVM_poly, mlp]:
156.         print("CLASSIFIER", clf)
157.         print(ml_pipeline(features, target, clf))
158.         plt.show()

```



## REGRESSION.py

Script for performing regression in python. Returns RMSE test and training scores.

```

1. import numpy as np
2. import matplotlib.pyplot as plt
3. import pandas as pd
4. from scipy import stats
5. from sklearn import linear_model, metrics
6. from sklearn.linear_model import Ridge
7. from sklearn.cross_validation import train_test_split
8. from sklearn.model_selection import LeaveOneOut, cross_val_score, KFold, ShuffleSplit, cross_val_predict, learning_curve
9. from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error
10. from sklearn.preprocessing import PolynomialFeatures, normalize, StandardScaler
11. from sklearn.svm import SVR
12.
13.
14. #Import data file from excel
15. excel_file = 'co2_measurements.xlsx'
16. #Convert data to panda dataframe, skip two first rows
17. measurements_raw = pd.read_excel(excel_file, "Sensor_Responses", skiprows=2)
18. #Drop columns with autogenerated data
19. measurements_drop = measurements_raw.drop(columns=['Unnamed: 0', 'Earth.1', 'Air.1', 'Difference.1', 'Unnamed: 16', 'OK?', 'Sample Number'])
20. #Drop measurements with two or more missing columns and take the absolute value of the rest
21. measurements = measurements_drop.dropna(thresh=2).abs()
22. #Standard scaler initializer
23. sc = StandardScaler()
24.
25. held_out1_ = [[-37903, -13031, -7362, -2427, -2955, -15761, -54334]]
26. held_out2_ = [[-21094, -22111, -10072, -2591, -2801, -24575, -100092]]
27. #Splitting data into different weeks
28. #WEEK 44s
29. ts = measurements.loc[:, 'Difference': 'PVA']
30. print(len(ts.Difference))
31. ts1 = measurements.loc[0:49, 'Difference': 'PVA']
32. #WEEK 44-45 gas measuring tube
33. ts2 = measurements.loc[50:70, 'Difference': 'PVA']
34. #WEEK 45-46 lowered in soil
35. ts3 = measurements.loc[72:104, 'Difference': 'PVA']
36. #TODO: make four loop for plotting, iterate through list of dataframes
37. #WEEK 45-46 sampling with:
38. ts4 = measurements.loc[105:110, 'Difference': 'PVA']
39. #WEEK 45.46 cont samp
40. ts5 = measurements.loc[112:124, 'Difference': 'PVA']
41. #WEEK 50 bubblestone
42. ts6 = measurements.loc[128:175, 'Difference': 'PVA']
43.
44. #String containing all the measurements
45. time_series = [ts1, ts2, ts3, ts4, ts5, ts6]
46. #Function calculating absolute error in percentage between true and predicted values

```



```

47. def mean_absoulte_error_perc(y_true, y_pred):
48.     y_true, y_pred = np.array(y_true), np.array(y_pred)
49.     return np.mean(np.abs((y_true-y_pred)/y_true))*100
50.
51. #Regression for testing different models
52. def reg_model(model, x_data, y_data, test_size):
53.     """
54.     inputs:
55.     - model: the learning algorithm to be trained and predicted on
56.     - test_size: the size of samples (number) to be drawn from training set
57.     - x_data: features input
58.     - y_data: target input
59.     """
60.     #Splitting data into testing and training sets
61.     x_train, x_test, y_train, y_test = train_test_split(x_data, y_data, test_size=test_size, random_state = 42)
62.     #fitting the machinelearning model to the data
63.     model = model.fit(x_train, y_train)
64.     #Making a prediction with the fitted model on the test data
65.     predict= model.predict(x_test)
66.     #Calculating a r2 score based on the actual and predicted data
67.     r2 = r2_score(y_test, predict)
68.     #Calculating RMSE score based on the actual and predicted data
69.     RMSE = np.sqrt(mean_squared_error(y_test, predict))
70.     #if the training and test score has large difference the model is overfit
71.     print("R2 score",r2)
72.     print("MAPE", mean_absoulte_error_perc(y_test, predict))
73.     #Printing testing and training score, if they differentiate a lot you are over fitting
74.     print("test score", model.score(x_test, y_test))
75.     print("train score", model.score(x_train, y_train))
76.     print("PREDICT 1", model.predict(held_out1_))
77.     print("PREDICT 2", model.predict(held_out2_))
78.     #printing root mean squared error
79.     print("RMSE",RMSE)
80.
81.     return
82.
83. #Initializing the different regression models
84. #CHANGE parameters for tuning
85. regr = linear_model.LinearRegression()
86. lasso= linear_model.Lasso(alpha = 0.0001, fit_intercept = True, normalize=False, max_iter =100) #l1 regularization, automatic feature selection
87. ridge = Ridge(alpha = 0.01) #l2 regularization
88.
89. #Function that shows the distribution of data
90. def distribution(data, transformed = False):
91.     """
92.     Visualization code for displaying distribution of the data
93.     inputs:
94.     - data: data that is going to be visualized
95.     """
96.
97.     # Create figure
98.     fig = plt.figure(figsize = (11,5));
99.
100.     # Skewed feature plotting
101.     for i, feature in enumerate(['Difference']):
102.         ax = fig.add_subplot(1, 1, i+1)
103.         ax.hist(data[feature],bins = 40, color = '#4682B4')
104.         ax.set_title("%s Feature Distribution"%(feature), fontsize = 14)
105.
106.         ax.set_xlabel("CO2 concentration diff")
107.         ax.set_ylabel("Number of Records")
107.     # Plot aesthetics
108.     if transformed:

```





```
109.         fig.suptitle("Log-
transformed Distributions of Continuous Census Data Features", fontsize = 16, y = 1
.03)
110.         plt.xticks(np.arange(100, 600, step = 50))
111.         else:
112.             fig.suptitle("Skewed Distributions of Continuous Census Data Feature
s", fontsize = 16, y = 1.03)
113.             plt.xticks(np.arange(100, 600, step = 50))
114.             fig.tight_layout()
115.             fig.show()
116.
117.         #Show distribution of data
118.         distribution(measurements)
119.         plt.show()
120.
121.         #Pick X and y data (features, target)
122.         X = ts.loc[:, 'PEO':'PVA']
123.         #Normalize x data row wise
124.         X_norm = normalize(X, axis = 0)
125.         X_test = X_norm[6]
126.
127.         y = ts['Difference']
128.         y_res = y.values.reshape(-1,1)
129.         #normalize y data column wise. .ravel() makes a np array
130.         y_norm = normalize(y_res, axis = 1).ravel()
131.         #Loop thorough models in list and test them through the regeression model fun
ction
132.         for model in [regr, lasso, ridge]:
133.             print("MODEL", model)
134.             reg_model(model, X, y_res, 0.40)
135.             plt.show()
```



## OT\_sampling\_jupyter.py

Sampling script for the opentron, ran in jupyter notebook.

The sampling script defines the path of the robot arm, sampling every other vial as earth and air. A rotation argument is utilized to avoid puncturing the same spot twice. Timers in the code should be set to match the timers in the Arduino code. Path taken by the robot is outputted in a .txt file.

```
1. #only needs to run once
2. get_ipython().system('pip install --upgrade opentrons')
3.
4.
5. # In[5]:
6.
7.
8. #Create container, only needs to be ran once on new computer
9. containers.create(
10.     '5x3_CG',      #name
11.     grid = (3, 5), #(columns, rows)
12.     spacing = (27.5, 23.5 ), #distances (mm) between each (column, row)
13.     diameter = 20.74,      #diameter (mm) of each well on the plate
14.     depth = 8      #Depth (mm) of each well on the plate
15. )
16.
17.
18. # In[1]:
19.
20.
21. from opentrons import instruments, robot, containers
22. from opentrons.util import environment
23.
24. robot.connect(robot.get_serial_ports_list()[0])
25. environment.refresh() #point jupyter to your apps calibration file
26.
27.
28. # In[14]:
29.
30.
31. #robot._driver.send_command('G28.2 Z')
32. #robot.resume()
33. robot.home()
34. #robot.move_head(z = 1)
35. #syringe.reset()
36.
37.
38. # In[13]:
39.
40.
41. import csv
42.
43.
44. #Variables, start theta at 0 degrees
45. theta = 0
46. time = 0
47. h = -0.7
48. #Create instrument
49. syringe = instruments.Pipette(
50.     axis='b'
51. )
52.
```



```

53. #Creating container plates
54. air_plate = containers.load('5x3_CG', 'A1') #Load container and its position
55. earth_plate = containers.load('5x3_CG', 'B1') #Load container and its position
56.
57. #for name, container in robot.get_containers():
58. print("Container: ", air_plate.get_type())
59.
60.
61.
62. #Function for sampling, delays for purging implemented
63. def sampler (plate, theta):
64.     for i in range(len(plate.wells())):
65.         robot._driver.send_command('G28.2 Z')
66.         syringe.delay(seconds=2)
67.         robot.home('b')
68.         robot.comment('move')
69.         #Move syringe to the first position of the plate, direct = direct, arc = ob
        st avoidance
70.         #move around a circle with radius (r) and theta (degrees)
71.         well_edge = plate.wells(i).from_center(r = 0.12, theta = theta, h = h)
72.         destination = (plate.wells(i), well_edge)
73.         syringe.move_to(destination, strategy = 'arc')
74.         #CHANGE delay if necessary
75.         syringe.delay(seconds=5) #Delay so position is set before inserting needle
76.         print("sample taken")
77.         #Lift needle and purge
78.         #robot.move_head(z=)
79.         # home z axis only
80.         #home b to tell arduino that the new sample is taken.
81.         print("changing air supply. ")
82.         # CHANGE delay to increase purge time
83.         syringe.delay(seconds=5) # pause seconds
84.         robot.comment('finished current plate')
85.         #theta += 0.785 #move penetration point with 45deg for each function call.
86.         return
87.
88.
89. #robot starts o theta = 0.314
90. #If there area ny warning reset and go home
91. if robot.get_warnings():
92.     print("WARNING", robot.get_warnings())
93.     robot.reset()
94.     robot.home()
95. else:
96.     #Start sampling
97.     robot.comment('Starting')
98.     robot.home('z')
99. #Sample 15 times the same rack
100.     for i in range(15):
101.         sampler(earth_plate, theta) #Air plate sampling
102.         theta += 0.785
103.         #Go home robot
104.         robot.home()
105.         #print out robot path
106.         commands = robot.commands()
107.         #Write all commands to txt file, can be used as error report.
108.         with open("output.txt", 'w') as resultFile:
109.             wr = csv.writer(resultFile)
110.             wr.writerow(commands)
111.
112.
113.
114.     # In[27]:
115.
116.

```



```
117.     robot.reset()
118.     robot.home()
119.     #Restart entierly by restarting kernel
120.
121.
122.     # In[5]:
123.
124.
125.     """
126.     Returns:
127.     axis_homed - axis that are currently in home postion
128.     switches - end stop switches currently hit
129.     steps_per_mm - steps per millimeter calibraiton values for x and y
130.     """
131.     robot.diagnostics()
```



## OT\_sampling.py

Calibration script for the OpenTron. Should be uploaded in the OpenTrons 2.0 GUI, and calibrate the plates.

```
1. from opentrons import instruments, robot, containers
2. import csv
3.
4. #Variables, start theta at 0 degrees
5. theta = 0
6. time = 0
7.
8. #Create instrument
9. syringe = instruments.Pipette(
10.     axis='b'
11. )
12.
13. #Creating container plates
14. air_plate = containers.load('5x3_CG', 'A1') #Load container and its position
15. earth_plate = containers.load('5x3_CG', 'B1') #Load container and its position
16.
17. #for name, container in robot.get_containers():
18. print("Container: ", air_plate.get_type())
19.
```

## Pump\_and\_valve\_circuit.ino

Script for the pump and valve circuit used on the Arduino. The script turns the pumps on and off based on timers set in the code. Valves are also opened and closed by the script to direct the flow of air either to the openTron or to the sensor. Timers must be set in accordance to the sampling script for the openTron.

```

1. //Initialize pins on the arduino
2. int OT_PIN = 2;
3. int earth_valve = 6;
4. int air_valve = 9;
5. int ot_valve = 11;
6. int air_pump = 5;
7. int earth_pump = 10;
8. int light = 12;
9. //Initializa global variables
10. int open_valve = 255; //Send max analog signal out to open the transistor
11. int close_valve = 0; //min analog vlaue to close transistor
12. int i = 0; //Counter
13. int buttonState = 0; //start buttonstate as 0
14. String pump1 = ""; //Empty string for filling in pump name
15. bool start = false; //start with start == off
16. bool air = false; //start with air as false
17.
18. void setup() {
19.   //Start communication with serial port
20.   Serial.begin(9600);
21.   //DEfine pins as output or inpit
22.   pinMode(air_valve,OUTPUT);
23.   pinMode(earth_valve,OUTPUT);
24.   pinMode(air_pump,OUTPUT);
25.   pinMode(earth_pump, OUTPUT);
26.   pinMode(light, OUTPUT);
27.   pinMode(OT_PIN, INPUT);
28.   //Light to indicate that everything is OK is turned on
29.   digitalWrite(light, HIGH);
30. }
31.
32. void loop() {
33.   //Read button state
34.   buttonState = digitalRead(OT_PIN);
35.   //If button state high start program
36.   if(buttonState == LOW){
37.     Serial.println("Robot has started ");
38.     delay(500);
39.     start = true;
40.     while(start){
41.       if(air == true){
42.         //Feed the sampling function with input
43.         takeSample(ot_valve, earth_pump, "earth ");
44.         delay(5000); //CHANGE: When defining a field test edit this parameter for tim
ing
45.         Serial.print("air");
46.         air = false;
47.       }else{
48.         //While robot has started, engage pumps
49.         takeSample(air_valve, air_pump, "air ");
50.         delay(5000); //CHANGE: When defining a field test edit this parameter for timin
g
51.         Serial.print("earth");
52.         air = true;
53.       }

```



```
54. Serial.println("FINISHED");
55. start = false;
56. }
57. }else{
58. Serial.println("Waiting for activation to start");
59. }
60. }
61.
62. void takeSample(int valve, int pump, String type)
63. {
64. Serial.println("Starting " + type);
65. //Turn pump on
66. digitalWrite(pump, HIGH);
67. pump1 = String(pump);
68. Serial.print(pump1);
69. Serial.println("opening " + type);
70. Serial.print(valve);
71. //Open valve
72. analogWrite(valve, open_valve);
73. delay(25000); //CHANGE: When defining a field test edit this parameter for purging
74. Serial.println("closing " + type);
75. Serial.print(valve);
76. //close valve
77. analogWrite(valve, close_valve);
78. //Turn pump off
79. digitalWrite(pump, LOW);
80. }
```



## C.2 Data collected

Data collected from the greenhouse experiment.

For download of the excel file

[https://eduumb-my.sharepoint.com/:f/g/personal/krom\\_nmbu\\_no/EoeRjNSSmSlEkboAPAsYG1wB0oavZ5FAw1oV2Bc\\_exuoQA?e=exoejJ](https://eduumb-my.sharepoint.com/:f/g/personal/krom_nmbu_no/EoeRjNSSmSlEkboAPAsYG1wB0oavZ5FAw1oV2Bc_exuoQA?e=exoejJ)

Data is presented below:





week 44

Sample Number	CO2			CMUT Chemical Sensor Δf (Hz)						
	Earth	Air	Diff	PEO	P4VP	PMMA	OV25	REF	PEI	PVA
1										
2	790	554	237	-36077	-12689	-7170	-2350	-2750	-15403	-50542
3	789	552	237	-34216	-12203	-7023	-2251	-2720	-14944	-46832
4	796	570	226	-38630	-12631	-7157	-2072	-2824	-15601	-53047
5	824	500	324	-36752	-12543	-7228	-2256	-2898	-15464	-50764
7	831	532	299	-38766	-13741	-7806	-2753	-2857	-15480	-77808
8	768	489	279	-30127	-12901	-6474	-2404	-2253	-13203	-41095
9	784	514	270	-52902	-14927	-10436	-3575	-4203	-19355	-117362
10	729	535	194	-39059	-13068	-5676	-364	-1650	-13721	-73724
11										
12	740	546	194	-49317	-15912	-9762	-3202	-3857	-18612	-21559
13	730	577	153	-49008	-16023	-9915	-3277	-3862	-18594	-52904
14	730	510	220	-49169	-15595	-9813	-3157	-3837	-18542	-60959
15	702	584	119	-30694	-11477	-6741	-2518	-2448	-13209	-64035
16										
17	762	542	220	-38011	-13421	-7390	-2489	-2672	-15457	-50207
18	756	541	215	-38011	-13421	-7390	-2489	-2672	-15457	-50207
19	774	560	213	-37794	-13053	-7505	-2525	-2751	-15359	-49103
20	754	563	191	-40453	-13476	-7901	-2569	-2870	-16107	-52341
21										
22	951	705	246	-36155	-9971	-6303	-1513	-2590	-13853	-37144
23	931	650	281	-34703	-8893	-5759	-1398	-2474	-12721	-28187
24										
25	856	601	255	-24555	-9990	-4985	-1422	-1417	-11570	-528
26	819	650	169	-26548	-10398	-5430	-1703	-1676	-12592	-16544
27	677	458	220	-29493	-10978	-5870	-1824	-1932	-13572	-22475
28										
29	720	469	250	-45394	-13408	-8184	-2453	-3058	-17708	-22827
30	683	458	225	-49212	-13604	-8800	-2700	-3457	-18531	-21376
31	665	471	195	-29907	-8965	-5778	-1513	-1767	-13086	2714
32	676	550	126	-29907	-8965	-5778	-1513	-1767	-13086	2714
33	648	498	150	-22708	-8208	-4801	-1388	-1385	-11036	3674
34	666	547	119	-23773	-8578	-5554	-2113	-1231	-11522	-1444
35	675	556	119	-19616	-7965	-4507	-1396	-1219	-10489	22957
36	682	500	183	-20946	-8397	-4753	-1472	-1386	-10838	22025
37	711	487	224	-24862	-8972	-5318	-1544	-1795	-12428	-32038
38	712	467	245	-34626	-10034	-6230	-1629	-2312	-14376	-53450
39	665	471	194	-45790	-11144	-7614	-2043	-3032	-16668	-71617
40	670	478	192	-45411	-11671	-8061	-2227	-3293	-17304	-45230
41	676	490	186	-53001	-13792	-12784	-7410	-6627	-22088	-69459
42	641	457	184	-47265	-11822	-8160	-2394	-3542	-16264	-61019
43	631	454	177	-38168	-10637	-7063	-1996	-2757	-14481	-6799
44										



	45	642	449	193	-19306	-8360	-5026	-2032	-1773	-10411	2368
	46	662	454	208	-22088	-8475	-4954	-1318	-1623	-11116	-4830
	47										
	48	685	478	207	-21691	-8181	-4914	-1196	-1838	-11175	-25642
	49	670	469	201	-14873	-5244	-4186	-812	-1688	-9307	-1205183
	50	653	465	187	-28305	-8063	-6154	-1694	-2485	-13336	-22968
week 44-45	55	712	511	201	-36499	-9802	-7072	-1744	-2794	-15323	-10072
	56										
	57	680	500	179	-18033	-7099	-4224	-1370	-1231	-9768	-21675
	58	691	496	195	-24823	-8771	-5421	-1440	-1831	-12638	312656
	59	691	485	206	-40291	-10679	-7658	-2194	-2968	-16237	-58165
	60										
	61										
	62	694	511	183	-69729	-13584	-10682	-3391	-4670	-21152	-53490
	63	682	498	184	-25144	-7442	-5033	-1156	-1467	-11473	-42375
	64	695	523	172	-18893	-6686	-4465	-1394	-1370	-10087	-15209
	65	671	522	149	-16547	-6262	-4040	-1294	-1135	-9431	-719453
	66	691	514	178	-18508	-7188	-5396	-2483	-2056	-10397	-545557
	67	655	515	140	-16779	-6359	-4069	-1240	-1202	-9658	-561757
	68	691	506	185	-19921	-6838	-4547	-1335	-1431	-10570	-67852
	69	689	504	185	-20152	-6710	-4696	-1338	-1543	-10622	-62248
	70	709	520	189	-24840	-7583	-5374	-1551	-2014	-11962	-54100
	71	687	500	187	-39369	-8474	-6911	-1857	-2391	-14835	-36389
	72	672	486	186	-22292	-6107	-4507	-821	-1084	-10915	16515
	73	690	484	207	-20200	-5899	-4182	-1042	-904	-10217	-40422
	74	680	499	182	-22454	-6562	-4533	-1286	-1050	-11014	-49717
	75	685	501	184	-34320	-8566	-6373	-1815	-2121	-14667	-2529
	76										
week 45-46	1	1111	564	547	-24625	-6621	-4807	-1437	-1352	-11145	-35675
	2										
	3										
	4										
	5										
	6	1083	548	534	-35471	-9096	-6631	-1311	-1098	-13425	-21642
	7										
	8										
	9	954	507	446	-13179	-4696	-3840	-1542	-1098	-8969	-34576
	10										
	11	1081	619	462	-29260	-9458	-6714	-2226	-2047	-14713	25073
	12	985	532	453	-28556	-9734	-6818	-2227	-2001	-14863	-11195
	13	923	533	391	-18635	-6653	-6140	-3167	-2581	-11995	-26719
	14										
	15										
	16	945	525	420	-20875	-7165	-5093	-1756	-1458	-11736	98404
	17										
	18	963	522	441	-12319	-4411	-3894	-1091	-762	-9793	-24154



19	923	539	384	-11414	-4046	-4271	-1707	-1249	-9776	-34930
20	934	491	443	-13862	-5380	-4739	-1877	-1412	-10686	-41444
21										
22	941	519	422	-23399	-8159	-6486	-2234	-1991	-13830	-22426
23	880	555	325	-32420	-9627	-7151	-2203	-2241	-15048	-95507
24										
25										
26	913	627	286	-22898	-7317	-5753	-1930	-1705	-12658	-38071
27	846	568	278	-30682	-8361	-7083	-2223	-2047	-15045	545397
28	846	591	255	-26059	-7876	-6537	-2123	-1860	-13758	-362201
29	868	550	318	-27688	-8928	-6889	-2529	-2192	-14295	716936
30	853	556	297	-13673	-4923	-4940	-1766	-1412	-10762	-12376
31	826	517	309	-14180	-4604	-4545	-1499	-1173	-10377	-16727
32	816	512	304	-29708	-6707	-5496	-1535	-1624	-12620	-232960
33	760	535	225	-21856	-5730	-5499	-1614	-1521	-12236	180057
34	826	510	316	-25069	-5529	-7651	-3669	-3453	-16380	405678
35	941	512	429	-26352	-7749	-6704	-2050	-2002	-14468	598310
36										
37	949	511	438	-6952	-2570	-5090	-1676	-1465	-10663	-12315
38	943	516	427	-6794	-2645	-5032	-1697	-1502	-10739	-9423
39	925	658	267	-5226	-2557	-4806	-1694	-1442	-10162	-20876
40										
41	937	490	447	-13823	-3770	-6149	-2105	-2014	-12891	-29714
42	900	485	415	-13794	-3671	-6043	-2015	-1901	-12648	-26596
43	946	488	459	-16071	-3943	-6323	-2188	-2084	-13018	-27196
44	956	493	462	-14740	-3981	-6356	-2104	-2065	-13101	-29182
45	975	498	477	-16512	-4428	-6611	-2215	-2132	-13663	-32872
46	962	492	470	-17256	-4530	-6665	-2365	-2298	-14031	-36056
47										
48	971	495	476	-18846	-4853	-6861	-2295	-2219	-14223	-37367
49	978	491	487	-19261	-5041	-6978	-2374	-2290	-14485	-38118
50	980	492	489	-20009	-4927	-7054	-2414	-2350	-14609	-40968
51	944	493	451	-19563	-4969	-7072	-2421	-2376	-14538	-37589
52	993	522	471	-18736	-6319	-6882	-2295	-2211	-14344	-35320
53	1016	517	499	-19254	-6134	-6941	-2392	-2306	-14571	-38692
54										
1										
2										
3	830	499	330	-31278	-17563	-7828	-2296	-2450	-16678	-83207
4	802	506	295	-25624	-24224	-10587	-2638	-2940	-23968	-117773
5	858	506	352	-24672	-24339	-10989	-3082	-3299	-24809	-119569
6	844	491	353	-24551	-24314	-11364	-3376	-3487	-25676	-120442
7	885	516	369	-23530	-23556	-11093	-3167	-3306	-25549	-116003
8	822	513	308	-22986	-23133	-10915	-3154	-3251	-25417	-111910
9	838	517	321	-22186	-22690	-10783	-3190	-3281	-25331	-110342

week 50



11	800	509	291	-20546	-21985	-10288	-3056	-3177	-24823	-96435
12	767	505	262	-19810	-20895	-10378	-3104	-3253	-24425	-95794
13	740	511	229	-18042	-20320	-9547	-2507	-2689	-23205	-92034
14	774	515	259	-17710	-20378	-9705	-2871	-2947	-23199	-90098
15	830	517	313	-16995	-20309	-9445	-2481	-2653	-22692	-88812
16	722	511	212	-16081	-20073	-9300	-2512	-2652	-22306	-86535
17	789	562	227	-15835	-19992	-9213	-2564	-2683	-22208	-85731
18	775	567	208	-17813	-21182	-12016	-5624	-4945	-24647	-87316
19	804	548	256	-14624	-19877	-8982	-2407	-2528	-21526	-83471
20	740	556	184	-15055	-20018	-9204	-2495	-2726	-21804	-84005
21										
22										
23										
24										
25	751	519	232	-20947	-11870	-5816	-1408	-1911	-11380	-79239
26										
27										
28										
29										
30										
31										
32										
33										
34										
35										
36										
37	725	503	221	-19561	-11554	-10213	-1999	-2210	-22863	-84642
38	679	500	179	-20333	-12738	-10522	-2332	-2348	-22982	-88802
39	744	501	242	-19589	-13290	-10284	-2242	-2261	-22559	-88292
40	752	490	263	-17807	-13659	-9782	-1969	-2017	-21615	-86249
41	736	476	260	-17705	-14351	-9839	-2214	-2194	-21448	-85721
42	692	477	215	-17449	-14691	-9640	-2066	-2087	-21184	-86087
43	755	476	279	-17306	-15152	-10031	-2527	-2435	-21396	-86009
44	716	485	231	-16887	-15333	-9525	-2111	-2218	-20754	-84881
45	759	485	274	-14863	-15458	-9144	-2072	-2044	-19950	-81381
46	705	498	207	-14835	-15728	-9197	-2222	-2247	-19810	-79604
47	652	516	136	-13601	-15852	-8667	-1751	-1860	-19227	-79036
48	769	496	273	-13214	-15965	-8442	-1565	-1747	-18828	-77845
49	695	486	209	-12850	-16310	-8662	-1929	-2026	-18907	-76807
50	723	494	228	-11875	-16363	-8198	-1657	-1779	-18116	-74687
51										

## Appendix D: Operation manual

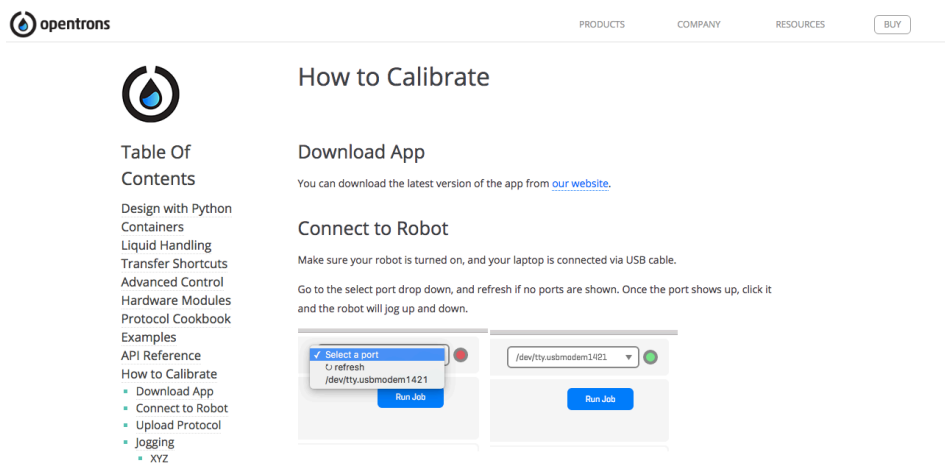
### Operation manual OpenTron and Arduino circuit

- How to calibrate the OpenTron autosampler
- Running the Jupyter notebook
- Running the Arduino circuit

### How to calibrate the OpenTron autosampler

- Go to the following web page for calibration procedure:
- <http://docs.opentrons.com/calibration.html>
- Use: OT\_sampling.py when asked for protocol file.
- Calibrate all slots listed.

IMPORTANT to disconnect the robot in the GUI before running the Jupyter notebook.



The screenshot shows the OpenTron website's 'How to Calibrate' page. On the left is a navigation menu with items like 'Table Of Contents', 'Design with Python', 'Containers', 'Liquid Handling', 'Transfer Shortcuts', 'Advanced Control', 'Hardware Modules', 'Protocol Cookbook', 'Examples', 'API Reference', and 'How to Calibrate'. The 'How to Calibrate' section is expanded, showing sub-items: 'Download App', 'Connect to Robot', 'Upload Protocol', 'Jogging', and 'XYZ'. The main content area has a 'Download App' section with a link to the website and a 'Connect to Robot' section with instructions to ensure the robot is on and connected via USB. Below the text is a screenshot of the robot's GUI, showing a 'Select a port' dropdown menu with 'COM1421' selected, a 'refresh' button, and a 'Run Job' button. The port dropdown is currently open, showing the selected port and a refresh button.

### Running the Jupyter notebook

#### Setup jupyter notebook on your computer

Setup video for Jupyter notebooks with python is found here:

<https://www.youtube.com/watch?v=HW29067qVWk>

In terminal: Navigate to the folder that contains the notebook code and open the jupyter notebook on your computer by running in the terminal:

```
jupyter notebook
```

The following screen should now open



```

jupyter OT_sampling Last Checkpoint: Last Friday at 8:18 AM (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help Trusted Python [conda root]
In [1]: 1 #!pip install --upgrade opentrons
        2 #only needs to run once
In [5]: 1 #Create container, only needs to be ran once on new computer
        2 containers.create(
        3     '5x3_CG', #name
        4     grid = (3, 5), #(columns, rows)
        5     spacing = (27.5, 23.5 ), #distances (mm) between each (column, row)
        6     diameter = 20.74, #diameter (mm) of each well on the plate
        7     depth = 8 #Depth (mm) of each well on the plate
        8 )
In [1]: 1 from opentrons import instruments, robot, containers
        2 from opentrons.util import environment
        3
        4 robot.connect(robot.get_serial_ports_list()[0])
        5 environment.refresh() #point jupyter to your apps calibration file

```

If it is the first time you run the notebook code on the computer do the following steps

- Run the block: `!pip install --upgrade opentrons`
- Run the block that creates the container: `containers.create()`
- Running blocks of code in Jupyter is done by hitting shift and enter simultaneously
- Run the connection block one time
- Home the robot by running the home block
- Run the sampling code
- The robot should now be sampling and saving the sampling procedure in a .txt file.

#### Tips&tricks

- To edit the place where the looping txt file is placed edit the following line of code specifying its path.
- Add custom blocks of code by adding more cell blocks, remember to define the type of well in the calibration script
- For adding a custom well see: <http://docs.opentrons.com/containers.html>
- For general questions about the OpenTron API see: <http://docs.opentrons.com/>

## Running the Arduino circuit

Install Arduino IDE on your computer, follow this tutorial

<https://www.arduino.cc/en/Main/Software>

Open the script: pump\_and\_valve\_circuit.ino

Your screen should look like this

```

int OT_PIN = 2;
int earth_valve = 6;
int air_valve = 9;
int ot_valve = 11;
int air_pump = 5;
int earth_pump = 10;
int light = 12;
int open_valve = 255; //Value between 0-255
int close_valve = 0;
int i = 0;
int buttonState = 0;
String pump = "";
bool start = false;
bool otr = false;

void setup() {
  Serial.begin(9600);
  pinMode(air_valve, OUTPUT);
  pinMode(earth_valve, OUTPUT);
  pinMode(air_pump, OUTPUT);
  pinMode(earth_pump, OUTPUT);
  pinMode(light, OUTPUT);
  pinMode(OT_PIN, INPUT);
  //Light to indicate that everything is OK
  digitalWrite(light, HIGH);
}

void loop() {
  //Read button state
  buttonState = digitalRead(OT_PIN);
  //If button state High start program
  if(buttonState == HIGH)
    Serial.println("Robot has started ");
    delay(500);
  }

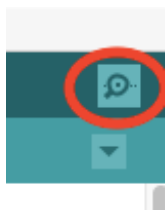
```

Connect the Arduino board to your computer

Verify and upload the code to the Arduino board



Open the serial monitor located right on the Arduino IDE



The serial monitor should now show

“Waiting for activation to start”

When the b switch is hit the valve circuit starts and the following message will show in the serial monitor:

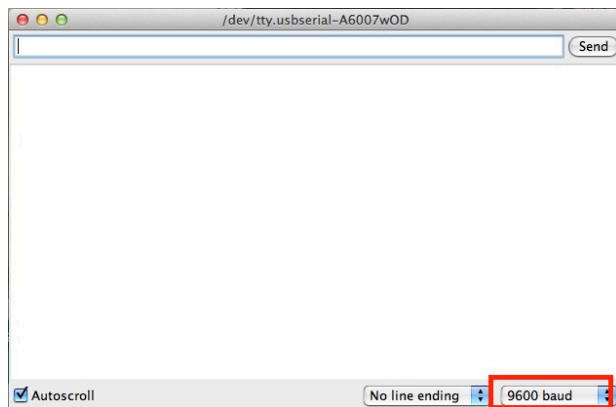
“Robot has started Starting pump x, opening valve x, closing valve x”

When the loop is ran every time the b switch is hit, if the b switch is not hit the circuit will go into standby.



## Tips&tricks

- If you get a connection error with the board go to tools → board and check if the board Arduino Genuino Uno is chosen, then go to tools → port and chose the port which the USB is connected though.
- If your serial monitor outputs alien language, check that the baud rate is set to 9600 baud







## Appendix E: Greenhouse test protocol

Equipment list:

CMUT sensor:

- Fitted on PCB.

Electronics:

- -60V power supply
- -15V power supply
- -5V power supply
- 3 V power supply
- Oscilloscope
- Frequency counter
- Bias voltage power supply
  - o +- 5 V dual power supply
- Bias voltage regulator
- Multimeter
  - o HP34401a

Controlling mass flow:

- Mass flow controller

Primary testbed

- Three pallets with soil and plants in closed greenhouse.

Analyzing the air samples

- Gas Chromatographer (GC).
- Air sampling bags.
  - o Multi-Layer Foil Gas Sampling Bags.
  - o Ca 20 Glass vials

Misc:

- Computer with Matlab and GPIB adaptor
- GPIB cable
- Serial cable
- 3V micro vacuum pumps with tubing and bubble stone.
- Nitrogen tank with 30 bar regulator.
- Power strips
- 2 Transformer (step down from 220-110V).

## Field test set-up

The test bed and pump set-up:

The pump setup will have one inlet of air from the greenhouse, and one inlet of air from the soil through a bubble stone that is placed into the soil of the plants. This system will have a pump capacity of 280ml/min, but will alternate between inlets and pump for 10 seconds at a time every 45 min. The pumps will direct the air currents to the sensors which is placed in a sensor box. During this testing there will be access to a gas tank with CO<sub>2</sub> so that the atmosphere in the greenhouse can be modified.

To avoid air pockets in the soil filling up with water the inlet must be placed in a way that does not direct water when irrigating to the inlet. This is illustrated in figure 1 how a pump set-up should look like if one would prevent the room around the stone from being filled with water and how the actual pump set-up looks like. The inlet with the bubble stone is placed at a 10 cm depth covered with soil.



**Figure 54** the placement of the pump inlet that is connected to the bubble stone surrounded by gravel to prevented saturated soil clogging the inlet.

In the specifications for the pump, the flowrate was set to 280ml/min and will pump air for 10 seconds every 45 minute so the sample will be about 40ml. The air will then flow over the CMUT sensor and is then collected in air sample bag (Multi-Layer Foil Gas Sampling Bags) or in a glass vial. The glass vials should be the best option.



Figure 2 Overview of the testbeds (from left to right): Box 1, Box 2 and Box 3.

Electronics:

Biased voltage regulator

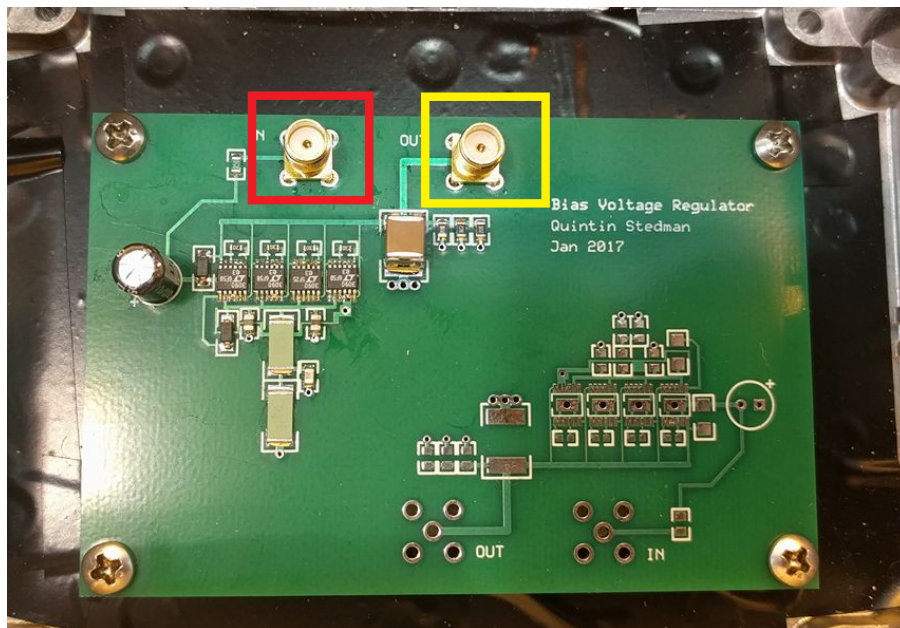
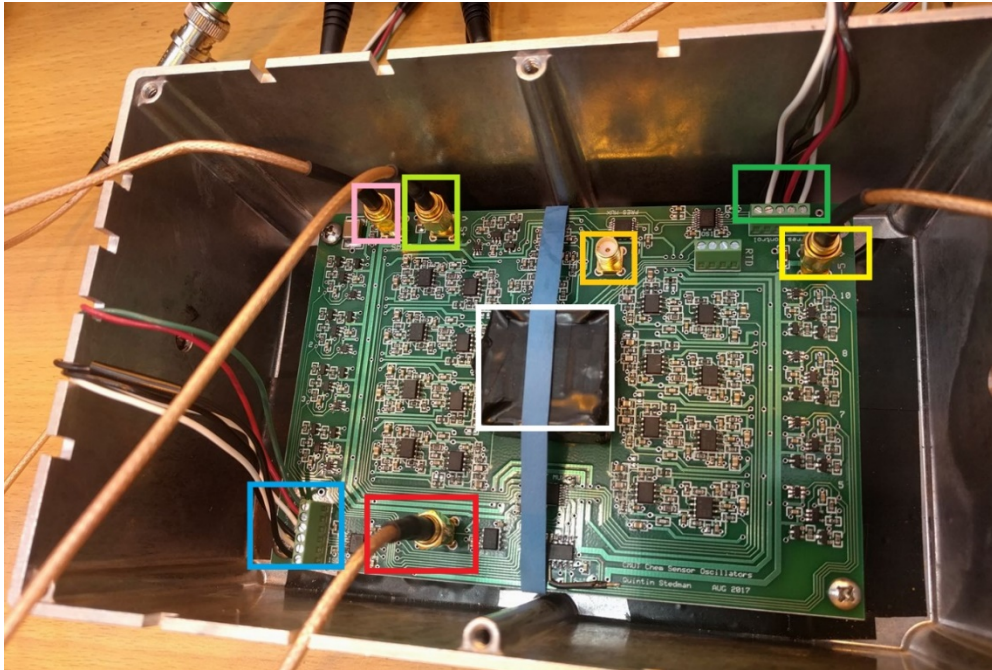


Figure 3 Overview of the bias voltage regulator which deliver a low noise stable 60V. Note the input voltage would give a +3V to the output (57V in = 60V out).



**Figure 4: Overview of the CMUT sensor set-up on PCB, sensor(White), +5V power supply (light green), bias voltage (pink), pressure sensor output (orange),from right to left (ground, +5 V digital output to the digital IO (1.4,1.5,1.6), Chemical sensor (blue (from top to bottom), Ground, +5V, 0.0, 0.1, 0.2, 0.3 ) and the – 5V (bright yellow).**



**Figure 5 Overview of the electronic set up (From left to right): Power supply to pump (earth), Frequency counter, CMUT sensor and voltage regulator (aluminum boxes), power supply to pump (air), power supply to the sensor set-up, the multimeter and computer where we run the Matlab Script.**

## Installing and setting up the equipment

The sensor will now run for five days (mon – fri) measuring every hour.

Each result will be uploaded to a joint dropbox account.

Initiating the sensor and the pump sequence:

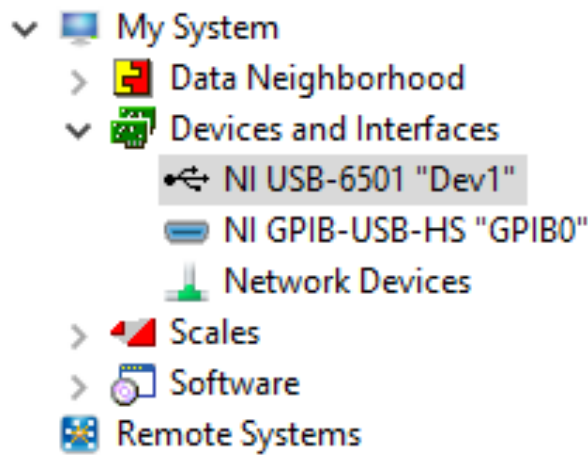
- Start-up the Computer
  - Prepare the matlab script
- Set the electrical equipment to “On”
  - The order doesn’t matter.
  - The power supply from “Stanford research systems” needs to be set to 63V in order for the CMUT sensor to get a 60V voltage.
    - Push “on” (far right button)
    - Then push “on” far left button.
    - Set the voltage by using the arrow and push enter for each 10 voltage to 50 voltage.
    - From 50V use single voltage slot to achieve a voltage of 63V.



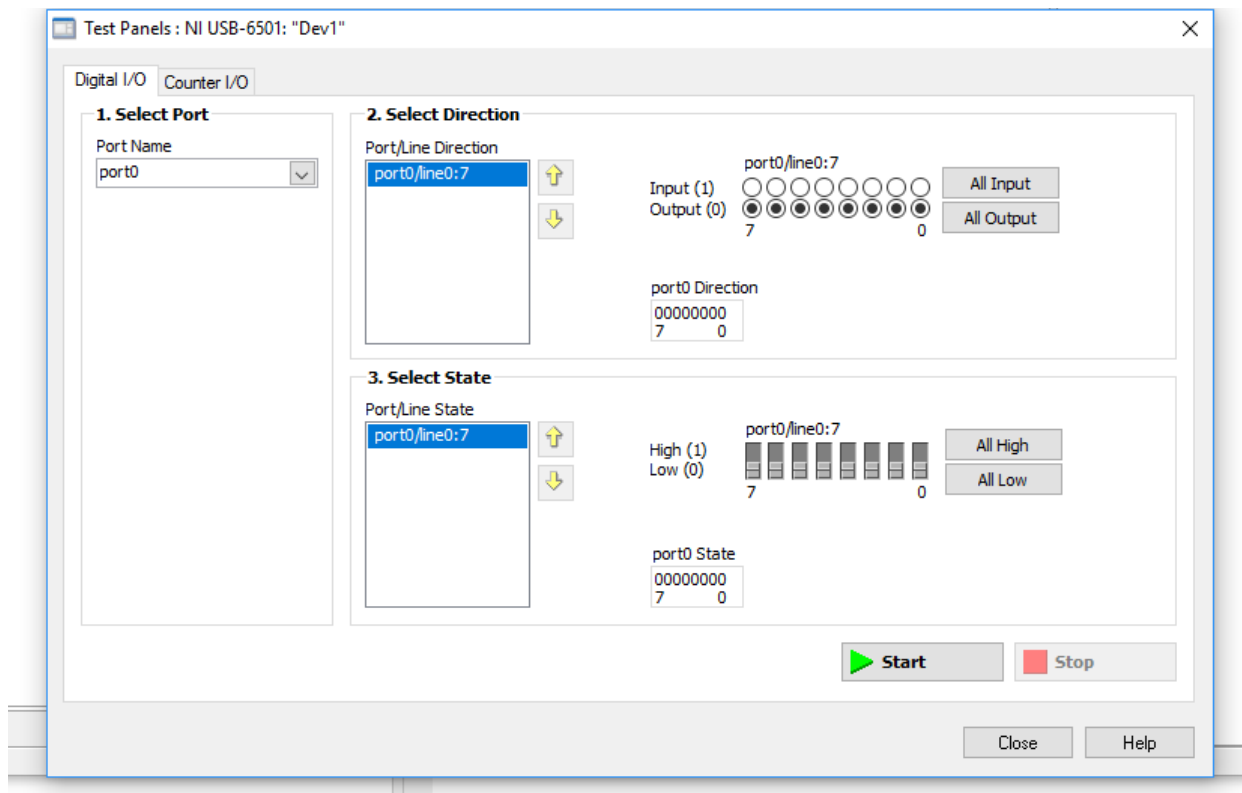
Figure 6: The SRS PS325 Power supply that supplies the CMUT sensor

- Never go beyond 63V on the sensor bias voltage.
  - Can cause the “membrane” to collapse.

- set the multimeter to four wire measurements.
  - Shift button -> 4W
- Test the sensor readings by running the program “NI Max”.
  - In “Devices and interfaces” ->



- Then enter the “Test panel” tab. This will display the following overview here one can display each of the polymer arrays (total 8, but in our setup 7) through the binary selection.



If the sensor works properly, each of the polymer array should display a frequency of 30 MHz in the frequency counter (remember to use channel 2 to display the frequency) as seen in the following figure.



Figure 7 Overview of the Keysight 53230A Frequency Counter

## Running the matlab script

Start Matlab.

In the command window:

Write soil then press Tab to generate a overview of different scripts that start single pump frequency as illustrated by the following pictures.

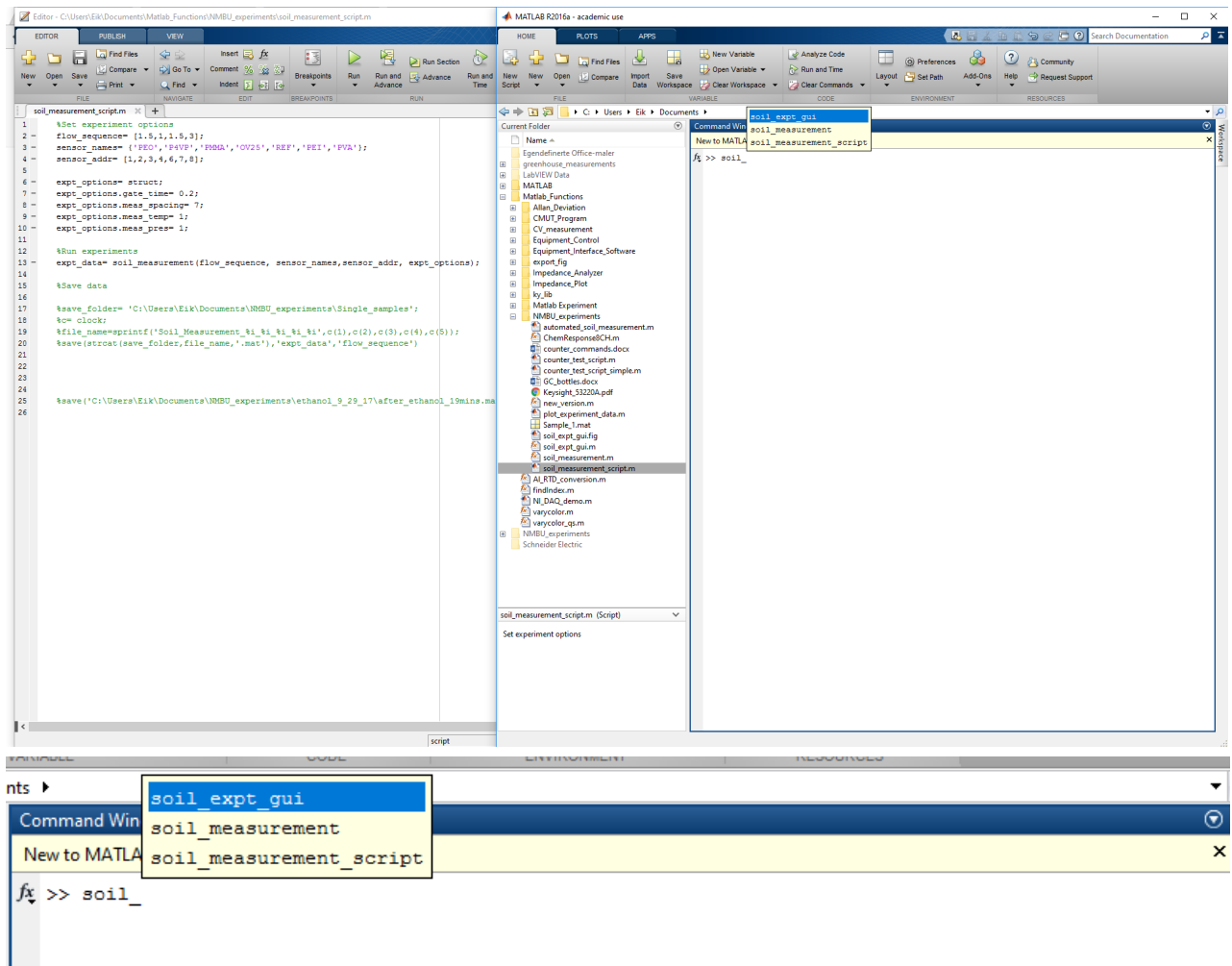


Figure 8 The matlab script and some basic commands to run the experiment



In order to initiate the test, call on the function “soil\_measurement\_script”. If successful the matlab command window will display the following

```
>> soil_measurement_script
Warning: A channel that does not support clocked sampling was added to the session. Clocked
operations using startForeground and startBackground will be disabled. Only on-demand operations
using inputSingleScan and outputSingleScan can be done.
Voltage of output 2 set to 3 Volts
Voltage of output 1 set to 0 Volts
Purging
Voltage of output 1 set to 0 Volts
Voltage of output 2 set to 3 Volts
Reference
Voltage of output 1 set to 3 Volts
Voltage of output 2 set to 0 Volts
Sample
Voltage of output 1 set to 0 Volts
Voltage of output 2 set to 3 Volts
Reference
Voltage of output 2 set to 0 Volts
Voltage of output 1 set to 0 Volts
```

Figure 8 Output from a successful iniation of the matlab script

And the pump sequence will be initiated.

In order to iniated the automatic sampling use the “automated\_soil\_mesurment” function the Matlab command window.

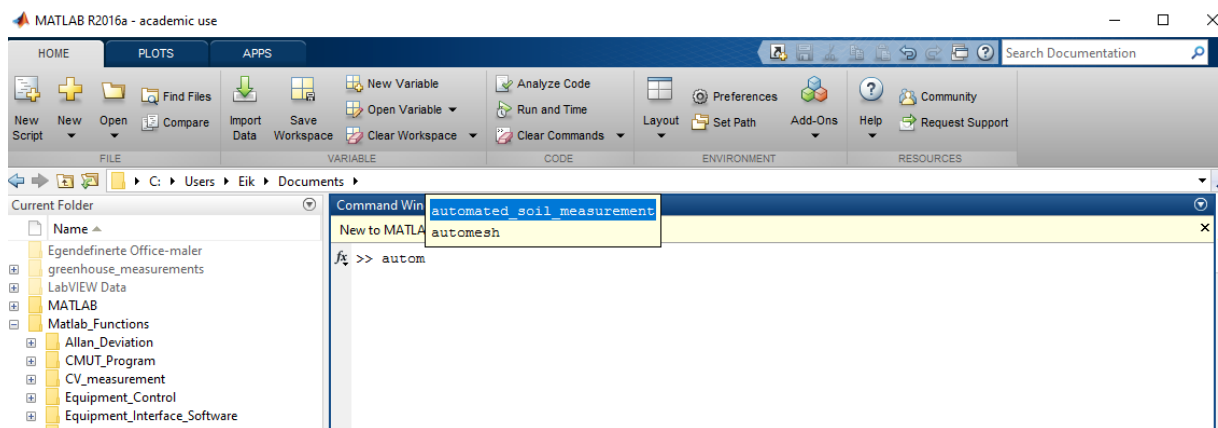


Figure 9 How to start the automated measurement sequence

This script has a runtime of 18 hours and will run the pumps each hour for 5 minutes.



### Problemshooting the sensor Set-up

If the sensor readings seems to be affected or greatly differs from previous readings try to

- Turn the power completely power off
- Tighten the cables running in to biased voltage regulator
- Turn on the power for the set-up
- Run “NI Max” to check the frequency over each sensor array.



## Experimental procedure

To verify the results from the sensor it is necessary to extract air samples parallel to the sensor. The samples will be then analyzed with a GC and used as a reference point. It will only be need for samples at crucial times when the plant nutrient and air supply is being altered. Most likely variations are

- Alter the air supply from greenhouse air to outside air.
  - Changing the position to different heights.
- Add different chemicals to the plant soil f.ex ammonia, fertilizer, Alcohol.
- Water the plants when measuring
- Add a chamber over the earth inlet.
- Increase the height of the soil layer in one of the boxes.
- Use different kind of tubing
  - PVC
  - PE
  - Silicon
- Gas stripper
  - Absorption column to remove the humidity from the air.
    - Placed at the inlet and outlet
    - Silica Gel or Zeolites (Inert material)

Preparation:

1. Obtain a reference sample and get this analyzed in the GC.
  - a. Air (from Greenhouse) and the soil (this is done).

In order to get good results it is important to sync the “manual” sampling with the automatic sampling.

Every hour the sensor will measure the air that is pumped over the sensor.

### Protocol - chem sensor measurement

By manually regulated the power supply(Keysight E3647A) for the two pumps one can preform sampling by adjusting the voltage on the different outputs (1 and 2).



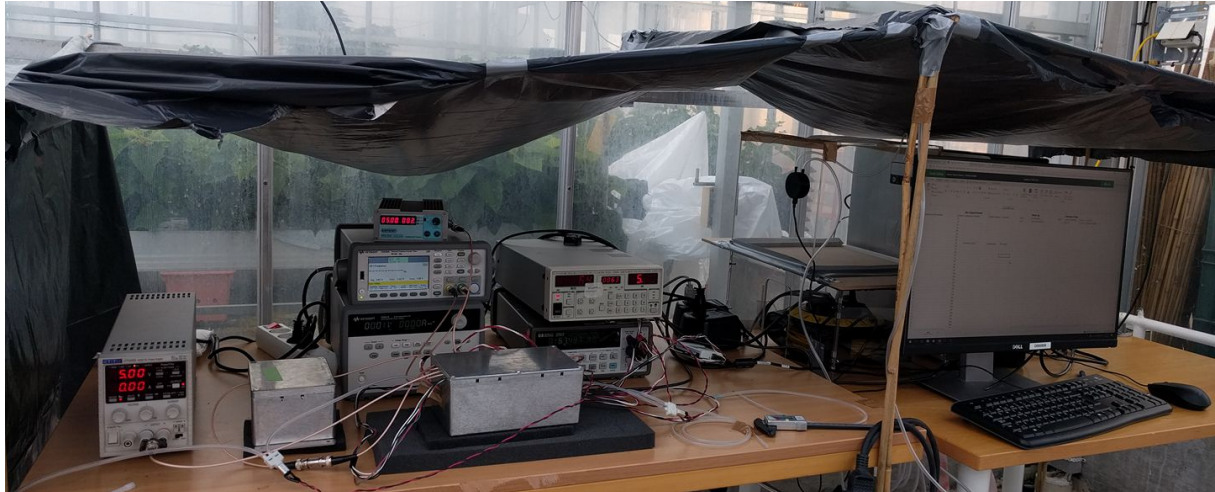
**Figure 10: By using output 1 and 2 you can regulate manually the micro vacuum pumps by applying 3 Volts, the picture to the right show how to collect samples for the GC.**

- The pump sequence during the “manual” sampling is:
- soil pump purge 10s (marked E#)
- soil pump collect 25 s
- air pump purge 10s
- air pump collect 25as

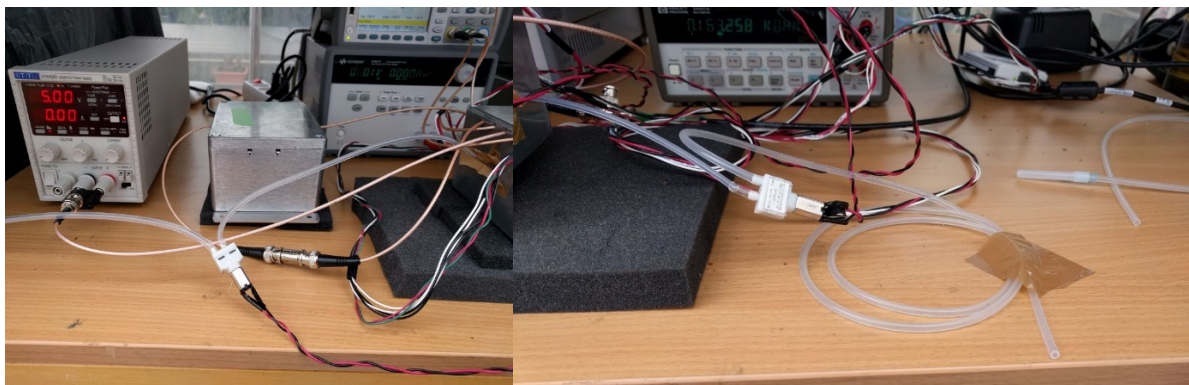
It is important to log the measurements and comment the alteration done in the experiment in the excel sheet “CMUT logging” in the field test folder in “onedrive”.

## Standard set-up

Overview of the experimental set-up (electronic plus testbed) which will be referred to “standard set-up” and will modified during the field trials.



**Figure 11 Overview of the test set-up**



**Figure 12: Soil sample pump (left) and Air sample pump (right)**

Overview of the different modifications applied during the field test.

### Pump holder

During the measurements the pumps are very sensitive to vibrations, a single touch will affect the measurements. Therefore, a holder for the pump was installed as illustrated in the following picture.

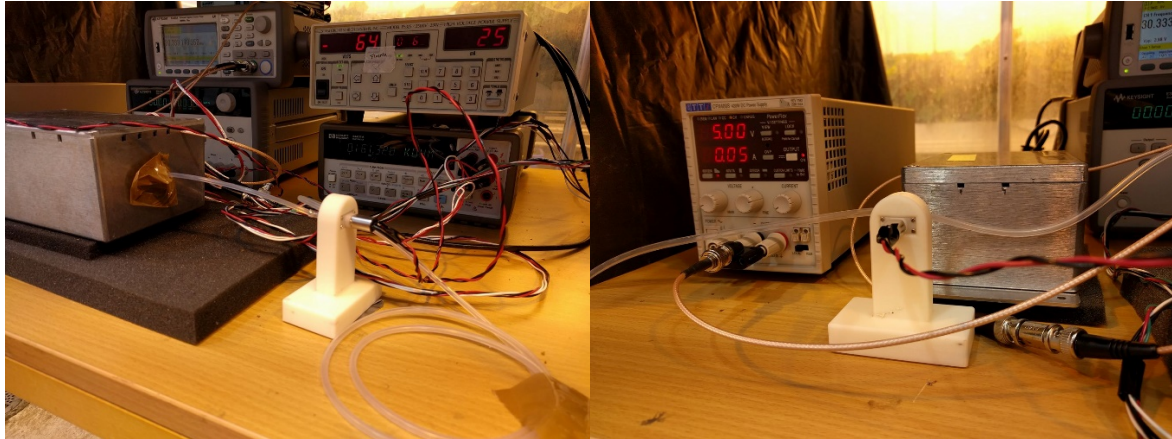


Figure 14 3D-printed pumpholders

### Outside air supply

In order to obtain outside air as a reference a hole was cut trough the Greenhouse wall as seen in the following picture.



Figure 16 The outside airsource inlet used as a reference

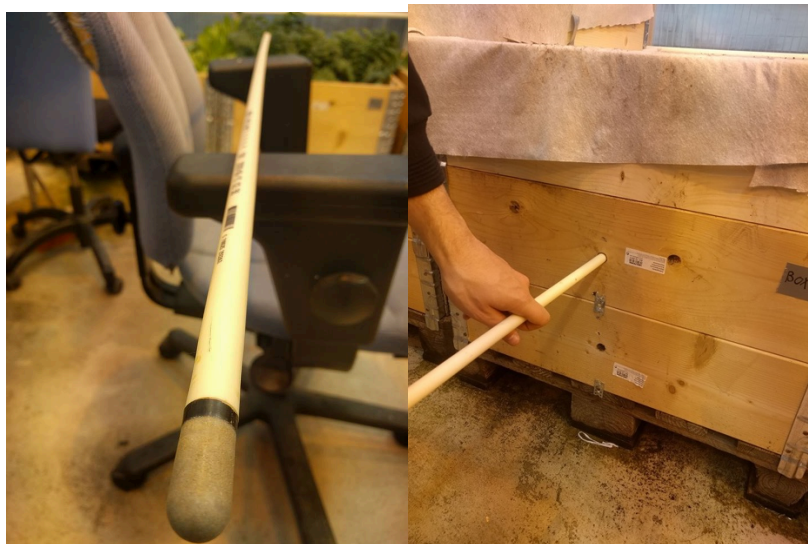
### Alterations done I Box 1 and 2

In order give the experiments a more realistic setting, box one and two was expanded in order to hold more soil in addition to several measuring points as seen in Figure. This will allow us some more degrees of freedom when performing the experiments.



**Figure 18** The modified boxed in order to accommodate more soil.

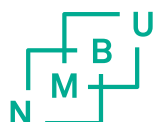
In this new set-up the project group will test out a “gas measuring tube” which is development for the single purpose of measuring gas in the soil. The tube consist of “silencer” (a product from Rexroth) attached to PVC tube. In the tube a silicon tube is coiled and is attached to a PE tube that’s works as an outlet. The idea behind this device is that the surrounding gas diffuses through the “silencer” and fills up the silicon tube coil. When the pump starts the sensor will measure the content of the tube and reduces the probability of drawing surface air through the set-up.



**Figure 19** The new measuring device for measuring gas in soil.







**Norges miljø- og biovitenskapelige universitet**  
Noregs miljø- og biovitenskapelige universitet  
Norwegian University of Life Sciences

Postboks 5003  
NO-1432 Ås  
Norway