Technical documentation

Henrik Eliasson
Yi-Ling Lo
Freja Nordgren
Alexandra Sandéhn

April 5, 2022

Version 0.2

NUTSENSE

Status

Reviewed	2020-01-05	Alexandra Sandéhn
Approved		

ACKNOWLEDGEMENTS

The completion of this project would not have been possible without all the help and support from all people involved in the project and the project course. Therefore, the members of the project group NutSense would like to take a minute and express our gratitude.

We would like to extend a special thank you to Donatella Puglisi, who was both examiner and customer in the course, for your inexhaustible encouragement and faith in the project from the very start. Even when NutSense felt stuck in the project process, your advice and reassurance gave us hope and energy to continue.

We would like so express our deepest appreciation to our supervisors, Guillem Domènech and Marius Rodner, who through an enormous commitment and interest have helped us through all stages of the project, including lectures, sensor construction, discussions and practical execution to mention a few. Without your help and knowledge we would not have been able to achieve the results that we did.

We also want to direct many thanks to our advisor Jens Eriksson, who always offered professional advice and thoughts concerning the project. We would also want to thank him for showing a large interest in the project idea and the results that followed.

In addition, we would also want to thank Svante Gunnarsson who had a very interesting and developing lecture/-workshop about the CDIO initiative and the LIPS model, Ida Iranmanesh who gave an inspiring guest lecture about her own career and Sally R and Filip Wiltgren for having a very helpful and powerful workshop on presentation skills. From these sessions, we all obtained knowledge and inspiration that we will carry with us forever.

Last but not least, we want to direct a collective thank you to everyone that attended our final presentation. We want to thank you for your interest in the project and its final results, for your questions and comments. It all means a lot to us.

NutSense

Project Identity

Examiner: Donatella Puglisi, Linköping university
E-mail: donatella.puglisi@liu.se

Customer: Donatella Puglisi, Linköping university
E-mail: donatella.puglisi@liu.se

Supervisors: Marius Rodner, Guillem Domènech
E-mail: marius.rodner@liu.se, guillem.domenech@liu.se

Advisor: Jens Eriksson
E-mail: jens.eriksson@liu.se

Participants of the group

Name	**Responsible**	**E-mail**
Henrik Eliasson	Responsible for data evaluation (DA)	henel953@student.liu.se
Yi-Ling Lo	Project leader (PL)	yillo603@student.liu.se
Freja Nordgren	Responsible for testing (TEST) and Quality check (QC)	freno283@student.liu.se
Alexandra Sandéhn	Responsible for documentation (DOC) and time report (TR)	alesa987@student.liu.se

CONTENTS

DOCUMENT HISTORY

Version	Date	Changes made	Sign	Reviewer
0.1	2021-01-05	First draft	AS	Alexandra Sandéhn
0.2	2021-01-19	Revised version	FN	Freja Nordgren

1 INTRODUCTION

In this document, all technical information about the sensor system project *NutSense* is presented. This project was a part of the Project course in applied physics, TFYA92 at IFM, Linköping university in the fall semester 2020. The goal of the project was to develop a sensor system for detection of peanut aroma. All work within the project has followed the CDIO project model.

1.1 Aim

The aim of this project is to fabricate a gas sensor system that can detect the odour of peanuts and warn when peanuts are detected. Ideally, the sensor is expected to warn the allergic people of the presence of peanut aroma before the allergy shock occurs.

1.2 Purpose

The purpose of this project is to investigateif it is possible to detect peanut odours by using gas sensors and what applications they are fit for.

1.3 Costumer

The customer of this project is Donatella Puglisi. Donatella is a senior lecturer in sensor and actuator systems (SAS), the department of Physics, Chemistry and Biology (IFM) at Linköping university.

Donatella Puglisi was awarded a MSc degree in Physics in 2005 and her PhD in Physics in 2009. She achieved her scientific habilitation in Applied Physics in 2016.

In 2019, Donatella Puglisi was awarded the national prize Åforsk Entreprenörstipendium. She is also the co-founder of the start-up DANSiC AB.

2 STATE-OF-THE-ART

In this section, a short summary of research and state-of-the art systems similar to the one fabricated within this project will be provided. The articles are published in the years 2017-2020.

2.1 Detection of peanut allergens in serum: circumventing the inhibitory effect of immunoglobulins (2020)

Peanut allergens can be detected in serum, but different assays give different results. Some assays had peaked and decreased after four to eight hours, whereas other still remained after two full days (48 hours). By using quantitative ELISA, Ara h 6 (peanut allergen) could be detected in most of the test subjects. In the study it was also concluded that heat treatment of the serum-samples enabled them to recover better from heat-stable immunoglobulin-bound food allergens [1].

2.2 Quality tracing of peanuts using an array of metal-oxide based gas sensors combined with chemometrics methods (2017)

In this study, both unshelled and peanut kernels were used. The setup of tracing models were based on 18 metal-oxide (MOx) gas sensors. The results of the classification were based on Linear Discriminant Analysis (LDA) and Support Vector Machines (SVM). The results of the study implied that the 18 MOX based gas sensors can be used, in combination with chemometrics methods, can be used for peanut quality detection. In addition to the method being effective, it is also defined as non-destructive [2].

2.3 Comparison of recovery and immunochemical detection of peanut proteins from differentially roasted peanut flour using ELISA (2019)

By using ELISA, peanut protein and flour was detected. Two protein assays and six commercial peanut ELISA kits were used. The roasted peanuts resulted in the highest peanut protein recovery by the ELISA kit and roasting of the flour resulted in a decrease in recovery. The protein and immunoassays alike implied that when roasting the peanuts/peanut flour, there was a decrease in peanut solubility. The study concluded that the commercial ELISA kits were not reliable in order to quantify the peanut presence in peanut flour at ≤ 25 ppm [3].

3 THEORETICAL BACKGROUND

In this section, theory related to the final system and its parts are presented.

3.1 What is peanut allergy?

Peanut allergy is an IgE-mediated disease. This means that contact with peanut protein (allergen) triggers an allergic reaction [4]. During exposure to the allergen, the peanut protein will bind to the IgE-antibodies produced by the immune system of the person exposed. This, in turn, triggers immune defences of the affected person, leading to an allergic reaction [5]. The symptoms of an allergic reaction varies on an individual level[4]. Some of the different symptoms are shown in Figure 1.

Figure 1: Signs and symptoms of food allergy reaction [6].

Typically, as can be seen in Figure 1, these symptoms include nausea, trouble with breathing and/or eczema [6].

Over the past decade, an increase in food allergy prevalence has been observed in Western as well as developing countries. It has been observed that, in children under the age of 5, the prevalence of food allergy varies in the range of 3.6 – 6.8% [7]. About 1% of these food allergies are related to peanuts, making this particular allergen amongst the most common ones [4]. In a report, covering 32 fatalities due to food allergen ingestion, peanuts and tree nuts accounted for 29 of the cases (approximately 90%) [8]. Similarly, solely in the US, approximately 6 – 8% of children younger than 4 and about 4% of the people older than 10 suffer from a food allergy [4]. In a review of 42 European studies published in 2000-2012, *Oral Food challenge*-reported prevalence of peanut allergy was 0.2%, whereas the self-reported lifetime prevalence of peanut allergy was higher, namely approximately 1.3% [7].

3.2 The emittance from peanuts

The aroma compounds that are emitted from raw and roasted peanuts respectively can be seen in Table 1.

Aroma compound	Aroma compound concentration from raw peanuts [$\mu g/kg$]	Aroma compound concentration from roasted peanuts [$\mu g/kg$]
Acetic acid	9176	3035
Hexanal	2734	838
4-vinylphenol	16	7814
Phenylacetic acid	109	2363
Phenylacetaldehyde	103	2427
4-vinyl-2-methoxyphenol	21	2017
4-hydroxy-2,5-dimethyl-3(2H)-furanone	15	1953
2-methylbutanal	17	971
(E,E)-2,4-decadienal	5.8	868
3-methylbutanal	14	637

Table 1: Aroma compound concentrations that are released from raw respectively roasted peanuts [9].

In Table 1 the highest concentration values and the corresponding values for the roasted/raw case are displayed, for comparison reasons. The three highest concentrations for raw peanuts are marked with yellow in the table and and the three highest concentrations for roasted peanuts are marked with orange. With this, it can be concluded that acetic acid has large emittance concentrations in both cases.

3.3 Data analysis

Data classification is a way to classify obtained data [10]. A classifier can be defined as an algorithm that maps the provided input data into groups [11]. It can viewed as a data management process that categorizes data into different groups. Through data classification, it is possible to sort data into this variety of groups according to a provided data set. Data classification is, in general, carried out by training data to scan and sort the provided data set into groups depending on a number of different parameters [10].

There are several different classification models to choose from, including *Logistic Regression*, *K-Nearest Neighbours*, *Decision Tree* and *Support Vector Machine*. The classification model uses the input training data to try to draw some kind of conclusion from it [11]. All classification models have both advantages and disadvantages, these (for some of the models) are shown in Table 2.

Classification model	Advantages	Disadvantages
Logistic Regression (LR)	useful when the goal is to understand the influence of several independent variables	predicted values must be binary, assumes independent predictors, assumes no missing data.
K-Nearest Neighbours (KNN)	simple to implement, robust, effective	high computation cost, needs value of K
Decision Tree (DT)	simple to understand, simple to visualise, small amount of data preparation, handles different types of data	risk for complex trees, can be unstable
Support Vector Machine (SVM)	effective, memory efficient	does not provide direct probability estimates, high cost for calculating probability

Table 2: Comparison of different classification models [11].

4 METHOD & PROCEDURE

In this section, the methods and the procedures followed throughout the project are described.

4.1 Reference measurements

The sensors will be described more closely in section 5.2. The two we used differed only in their sensing layer. The first sensor studied was based on epitaxial graphene (EG) while the other was epitaxial graphene decorated with platinum nanoparticles (Pt NPs).

For data driven modeling it is of interest to have as much data as possible from different sensors. To obtain enough raw data to make a worthwhile analysis while using a limited amount of sensors, we have used temperature cycles. The sensor cycled between four temperatures: $100°C$, $120°C$, $140°C$ and $160°C$. In this way, we achieved a virtual sensor array consisting of four virtual sensors running at different temperatures. A temperature cycle from our measurements of the sensor with Pt nanoparticles is displayed in Figure 2. It shows how the sensor resistance is changing based on what temperature we tell the heater to apply. We chose to apply each temperature for 40 seconds. This time was found to be long enough for the sensor to sufficiently stabilize at each temperature.

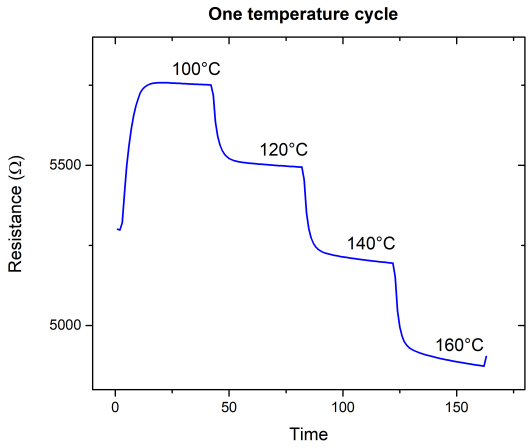


Figure 2: A temperature cycle.

The setup of reference measurements can be viewed in Section 5.4. Mainly, three different parameters are controlled: peanut weights, relative humidity, and temperature values of the temperature cycle for the sensor the operate. Values used are shown in the table below. Except for that the peanut weights must be changed manually by opening the 3D-printed peanut chamber, relative humidity and the temperature cycle can be set by the lab computer with specific software which will be introduced in the following section 5.5.

Parameters	Values
Peanut weights [g]	0.5, 2, 5, 10
Relative humidity [%]	5, 15, 20, 30, 40, 50, 60, 70
Temperature [$^{\circ}C$]	100, 120, 140, 160

4.2 Data analysis

To elucidate if peanuts and air could be distinguished a PCA analysis on our sensor data was performed. We used the PCA function in matlab to generate a score which later could be plotted. To use matlab's PCA function we first needed to extract the correct data from our measurements. To obtain the virtual sensors from the measurements performed, a temperature cycle was applied. With this, a value from each temperature plateau in each cycle to receive responses from four virtual sensors had to be sampled. This is displayed in Figure 3.

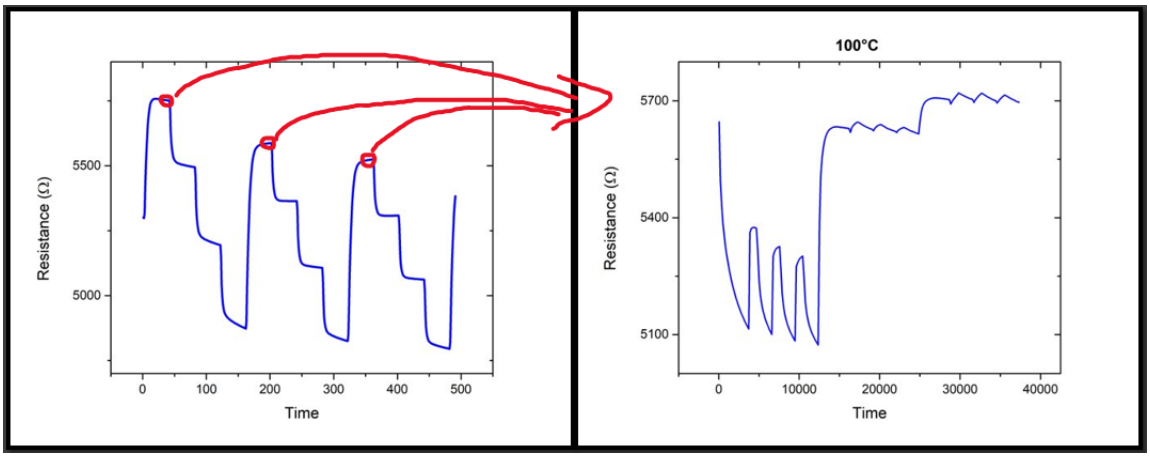


Figure 3: How sampled values at 100°C from the raw data form a response curve for a virtual sensor.

When the virtual sensors were extracted from the raw data, what features to look at had to be decided. We chose to only look at the mean values of the exposure plateaus. We also included some mean values of air to compare with. The mean values were normalized before they were used in the PCA.

The data classification was carried out through a number of steps. First, the mean resistance value from each temperature of the temperature cycle was calculated. Second, the mean and standard deviation value of these values were calculated in order to scale down the data to one dimension. Third, the normalized value of the dataset was calculated, creating an array of the mean values of all temperatures. Fourth, this array was turned into a column matrix and the mean value was double checked to be close to zero and the standard deviation to be one. Fifth, the normalized mean values was run through the decision tree classifier model and a decision tree was generated. By using this matrix and by specifying where the interval of air (in the matrix) started, a "decision surface" could be generated as well.

The data classification on the hybrid system experimental measurement was performed in-situ, while exposing the sensor to gases. A text-file of normalized resistance measurement values were transformed into an array of values. Next, the values of air exposure were separated from those of peanut exposure and bundled together at the end of the array. From this, a column matrix was generated. By specifying the interval of peanut- and air exposures, this matrix was then used to generate a decision tree and its corresponding decision "surface".

4.3 Implementation

To use the PCA function from matlab, a feature matrix must be introduced as input argument to it. This feature matrix was created by gathering all their normalized mean values from each virtual sensor in their own columns. We ended up with four columns since our temperature cycle included four temperatures. The function "*pca*" returns a score matrix where each column represents a principal component. Then, principal component 1 and 2 were plotted to yield the results, through which data distribution information was obtained. The results are which are discussed later.

In order to implement the training of the data classification from the reference measurments, Pyhton was used

and the code was written in Google colaboratory. Files containing data (generated from the PCA-implementation) was uploaded to this environment in order to be run through the classification model. These files were:

- 30RH_COI.mat
- 30RH_COI_MEAN.mat
- 30RH_COI_STD.mat

The code can be seen in Appendix B and the files containing all data can be viewed in Appendix F, G and H.

Through hybrid system experimental measurements, this implementation was revised and adjusted accordingly. This implemetation was also written in Python, using the Google colaboratory environment and used the following file:

- serialcopy.txt

which is a copy of the serial window of the Arduino measurements. The complete code for this can be found in Appendix C and the serialcopy data-file can be found in Appendix I.

The arduino code for determining whether peanut aroma is detected or not was implemented by using the decision tree generated by the training of the reference data. The environment for the implemetation was arduino software. The final code for this can be found in Appendix D.

5 PROTOTYPE OVERVIEW

This section provides an overview of the final prototype system. This overview covers the interior and exterior of the prototype and includes the demonstrator parts, the sensors, the electronics and the software of the total system.

5.1 Box

A design of the demonstrator box for 3D-printing is displayed in Figure 4. For design, the 3D-modelling software Fusion 360 has been used. The dimensions of the designed box are 10x15x20 cm and the lid consists of holes in a rectangular pattern to create an airflow towards the sensors. The wall thickness is 3 mm. For the lid and box to fit together, the lid consists of an extrusion with dimensions equal to the box inner side. The material used for printing is polycarbonate.

As the designed demonstrator box might have been to large for 3D-printing, the box in Figure 6 was used for demonstration with the speaker working as a lid.

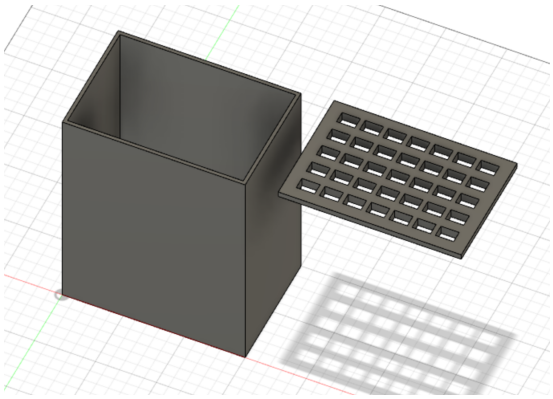


Figure 4: Design of the demonstrator box

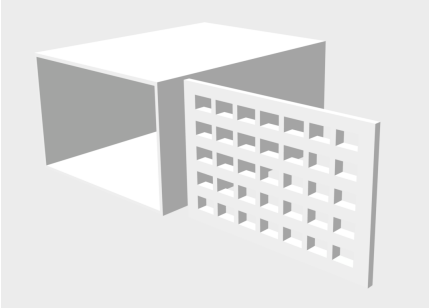


Figure 5: Demonstrator box for printing

5.2 Sensor

Sensors with three different sensing materials were prepared beforehand, including (1) pure graphene, (2) graphene decorated with Au-nanoparticles (NPs) and (3) graphene decorated with Pt-NPs. A sensor mounted on platform is displayed in Figure 7. The sensor with Au-NPs could not be used afterwards. Instead, mainly the sensor with Pt-NPs deposited on graphene and the sensor with pure graphene were used during reference measurements. The sensor has four main parts: a heater, sensing material, electrodes, and a platform to hold everything together. More details regarding fabrication of the sensor are provided in the next subsection.

Figure 6: Final demonstrator box

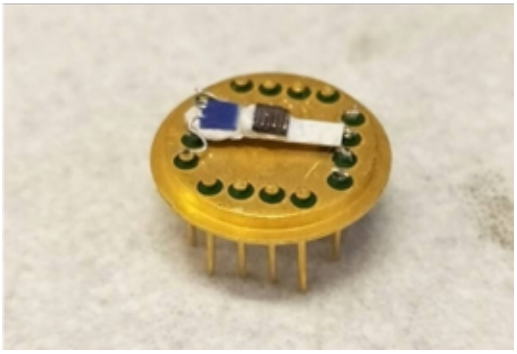


Figure 7: Sensor mounted on TO-8 package

5.3 Fabrication

The fabrication of the sensors was carried out through the following steps:

- Deposition of contacts on graphene by sputtering: (see in Figure 8)
- Heater welding: for the heater, not only the ceramic heater has been welded but also a Pt100 temperature sensor, which is ready to be connected with the temperature controller. In Figure 9 (a), the welding tool with a sharp end is shown. During the welding process, a current was applied through this sharp end on the heater wires, in order to secure them to the pins on the platform. In Figure 9 (b), the picture provides a closer look of the welding process under the optical microscope. One leg of the heater has been welded but the other one has not.
- Deposition of nanoparticles on graphene: for the sensing materials, Pt-NPs were deposited on graphene by dripping 99% liquid ethanol with the nanoparticles, a process referred to as “drop casting.” Since ethanol is volatile, it will soon be only Pt-NPs left on the graphene.

- Electrodes attachment: some tiny gold wires as electrodes were attached to the pins on the platform with the help of supervisors.

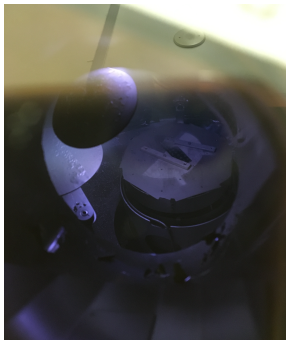
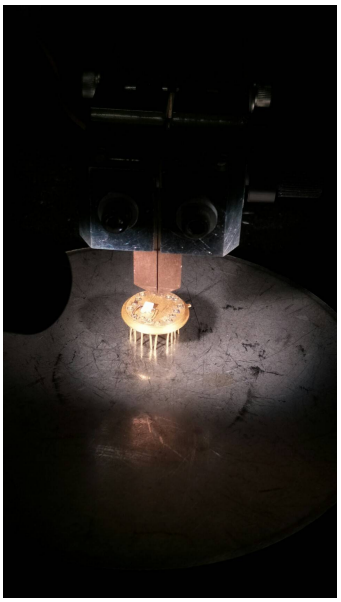
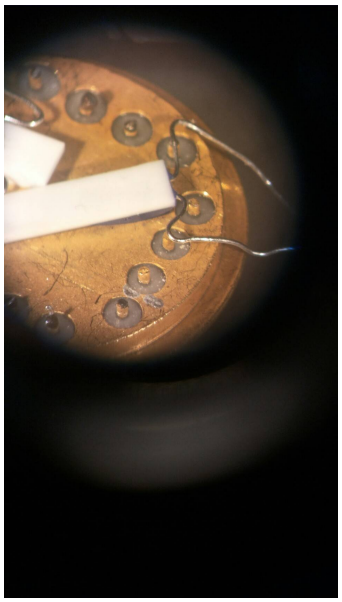


Figure 8: Deposition of contacts on graphene



(a) The welding position.

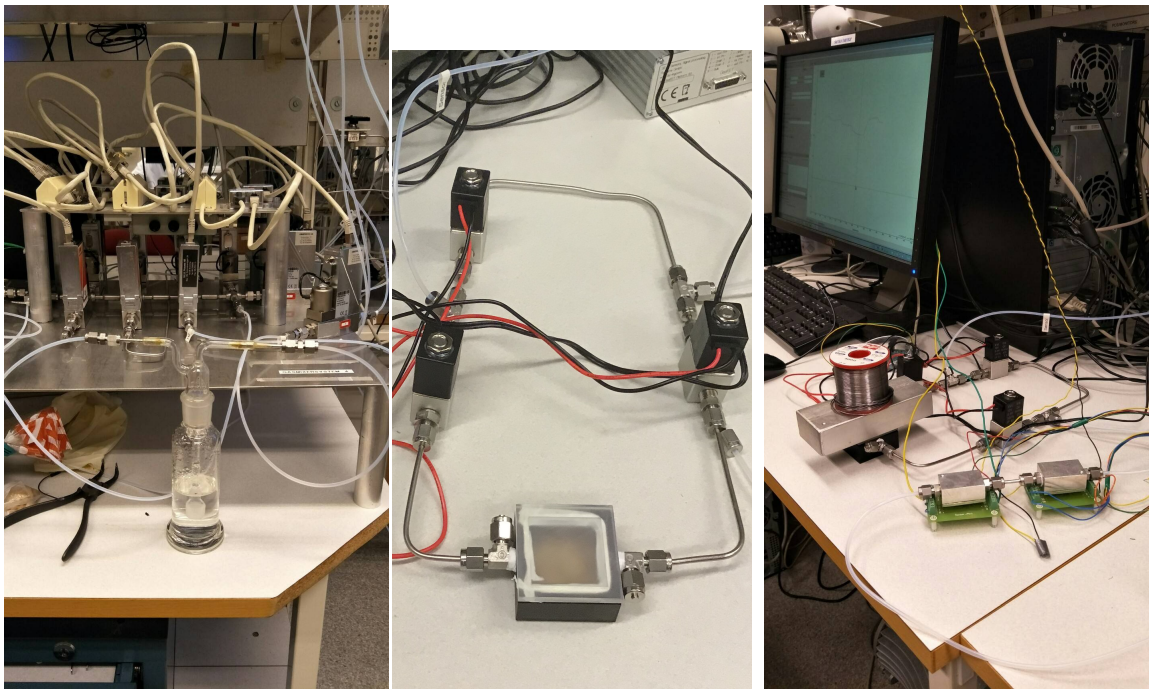


(b) A closer view under optical microscope.

Figure 9: The welding process of heater and heat sensor.

5.4 Electronics

Related setup images of the experimental measurements are shown in Figure 10. In order to build a virtual sensor array, both sensors were used to measure different weights of peanut under a temperature cycle and different percentages of relative humidity. Figure 10(a) is the bubbler for relative humidity control. Synthetic air flows through the transparent tubes to pneumatic valves, which are shown in Figure 10(b). The valves control whether there is air or peanut aroma interacting with the sensors. A 3D-printed peanut chamber and three valves are needed. The valve on the top left allows air to flow in to the sensors and the other two valves control the flow of peanut aroma. In Figure 10(c), there are two metal boxes, which are sensor chambers connected to the temperature controller and the valves. All the parameters for the reference measurements were set by using the programs in the laboratory computer.



(a) The humidity controller. (b) Peanut chamber and valves. (c) Reference measurements setup.

Figure 10: Experimental setup.

Figure 11 gives a closer look of the laboratory setup of the hybrid peanut sensor system. Beside all the other components that have been introduced above, there is a speaker connected to the arduino board to act as an alarm when the system detects peanuts.

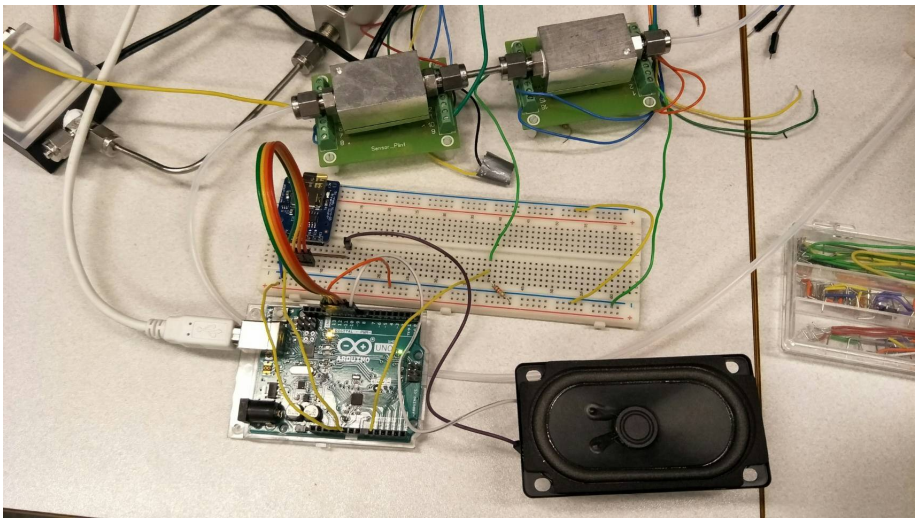


Figure 11: The laboratory setup of the hybrid peanut sensor system.

5.4.1 *The demonstrator circuit*

Figure 12 displays the complete schematic of our demonstrator wired up with the laboratory system.

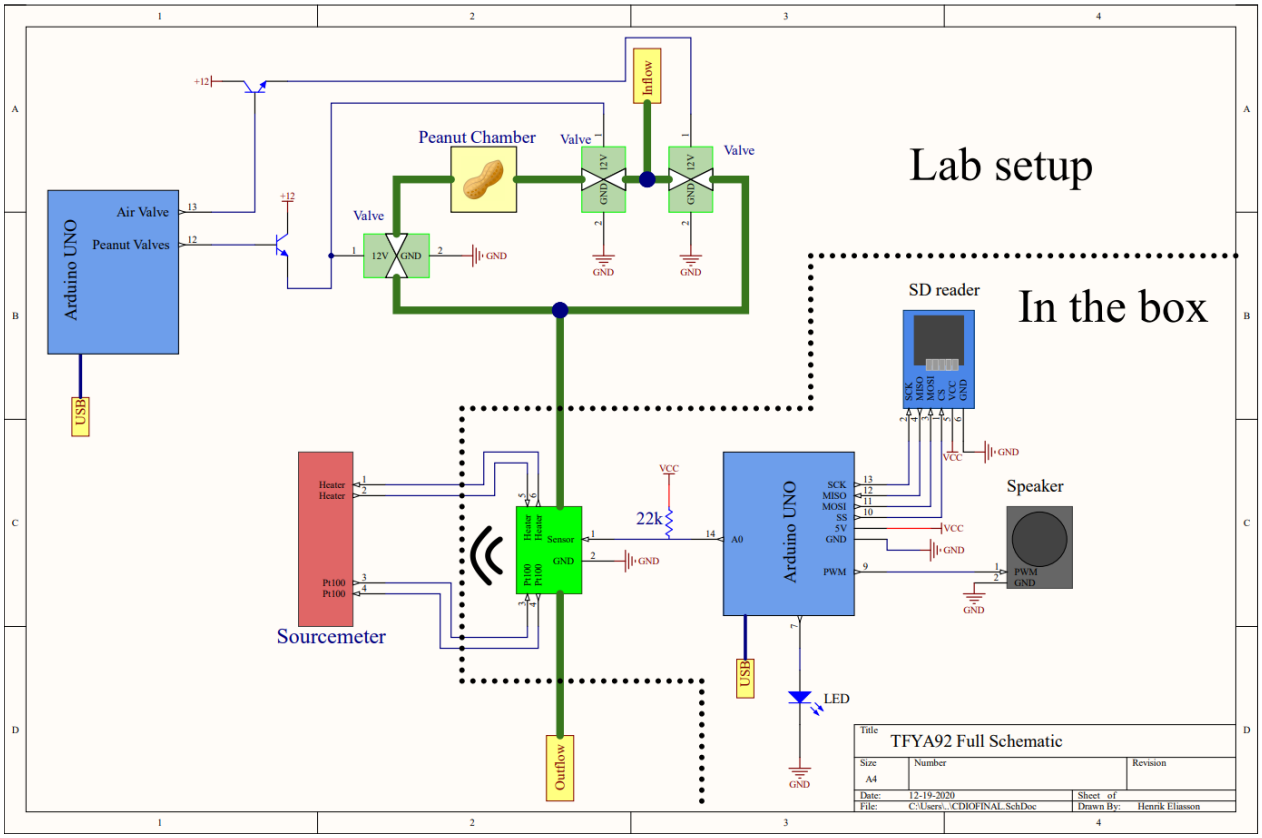


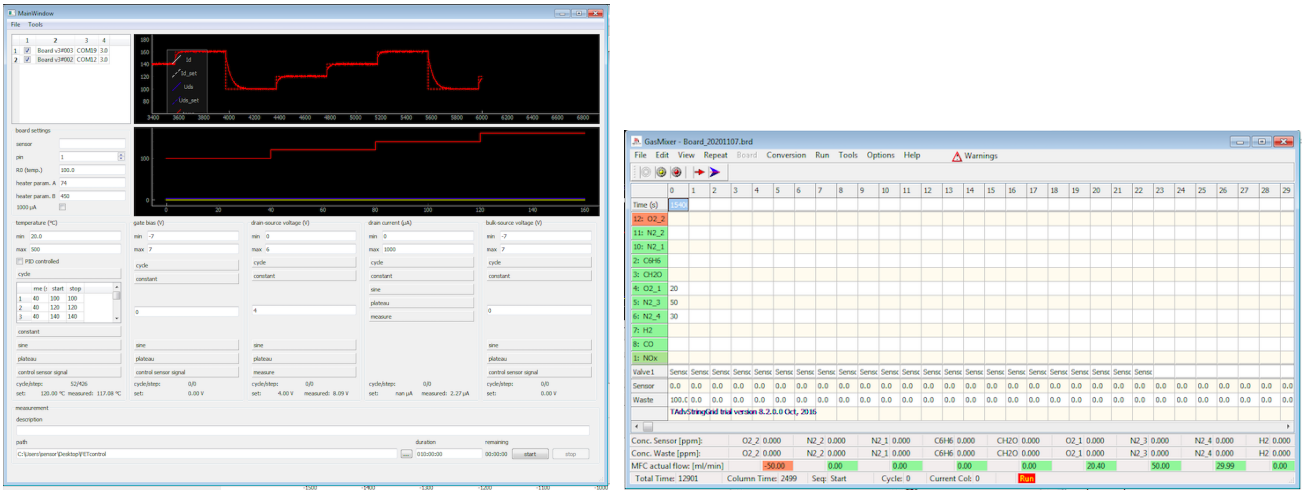
Figure 12: The demonstrator circuit. The green thick lines represent tubes where air is flowing to our sensor chamber (the lime green rectangle). The box that is referenced is the blue demonstrator box in Figure 22.

The micro-SD card reader is communicating with our sensor via SPI (serial peripheral interface). The speaker signal is a PWM output from the arduino controlled by the arduino software. To be able to switch between peanut contaminated air and clean air, the tubing was split and three valves were added. The valves control which way the air is flowing, either through the peanut chamber or not. The valves were controlled with a second arduino connected to the lab system. The arduino generates a 5 V digital control signal to a transistor which is connected to 12 V. If the transistor is opened or closed determines if the valve is open or closed. If the control signal is 5 V, the valve will open, and if it is 0 V, the valve will close.

5.5 Software

In order to control the laboratory system setup, several interfaces were used. A computer program of the laboratory computer was used for controlling the temperature and temperature cycles of the heater. In order to control the relative humidity (RH[%]), the program GasMixer was used. These interfaces are shown in Figure 13. For con-

trolling the valves of the system and the collection of data from the sensors, arduino software was used. These interfaces are shown in Figure 14. The complete programs can be seen in the Appendix D and E.



(a) Computer program for controlling the temperature.

(b) Computer programs for controlling the relative humidity.

Figure 13: Computer programs for controlling the temperature (a) and the relative humidity (b).

Figure 14: Interface for Arduino boards control.

6 MATERIAL

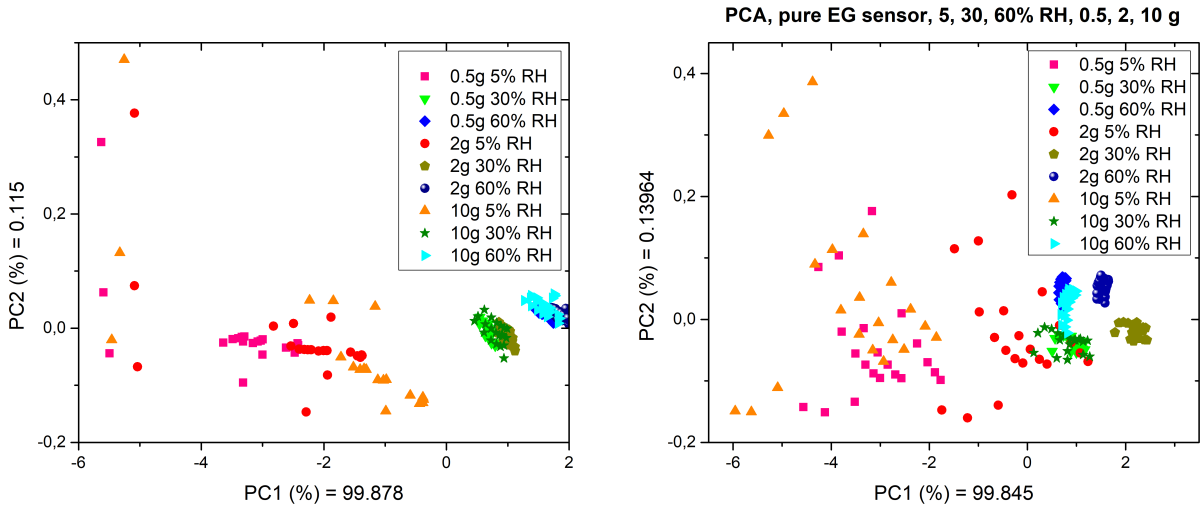
From the sustainability point of view, the components of the final product were chosen to be as recyclable as possible. For instance, the demonstrator box is a reused one from the laboratory, so are the platforms of both sensors. Also, the 3D-printed peanut chamber is made of plastic threads. If it should be disposed, this plastic material can still be melted to form another desired shape, which is perfectly corresponding with the sustainability goals.

7 RESULT

In this section, the results of the project are accounted for.

7.1 PCA

From the PCA analysis, which was performed on each sensor's data, the score plots displayed in Figure 15 was the result. The data contains information from measurements taken at different temperatures and relative humidities. It is clear that for the sensor with Pt NPs the data clusters depend on the relative humidity. The result is not as clean for the pure EG sensor where there is a bit of overlap between clusters.

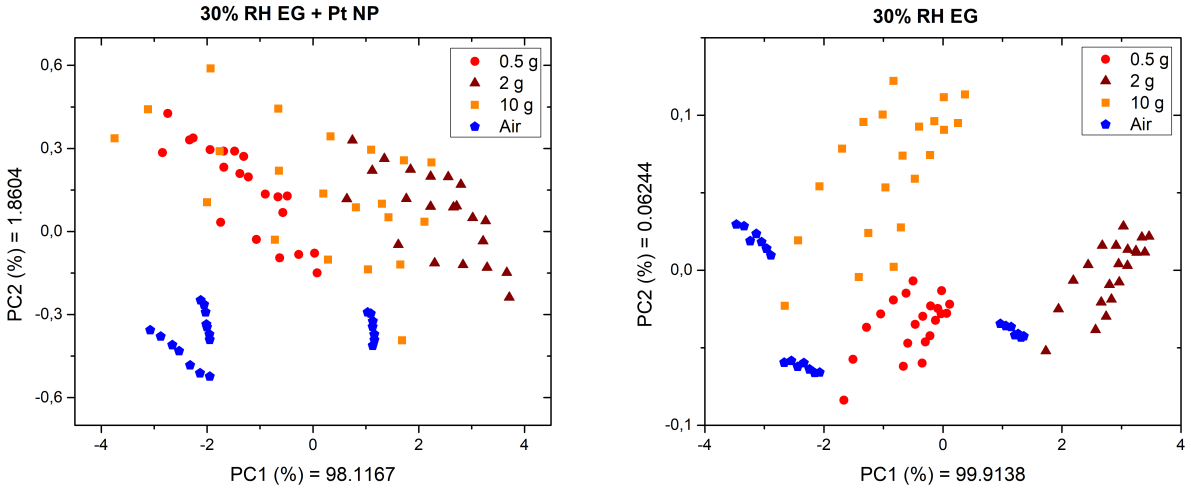


(a) PCA on data from the sensor with Pt nanoparticles.

(b) PCA on data from the pure EG sensor.

Figure 15: Separate score plots returned by PCA analysis on data from each sensor.

We chose to focus on the measurements taken at 30% relative humidity since it was the humidity closest to regular indoor climate out of the three we studied. A new PCA analysis for the 30% relative humidity gave the results displayed in Figure 16.



(a) PCA on data from the sensor with Pt nanoparticles.

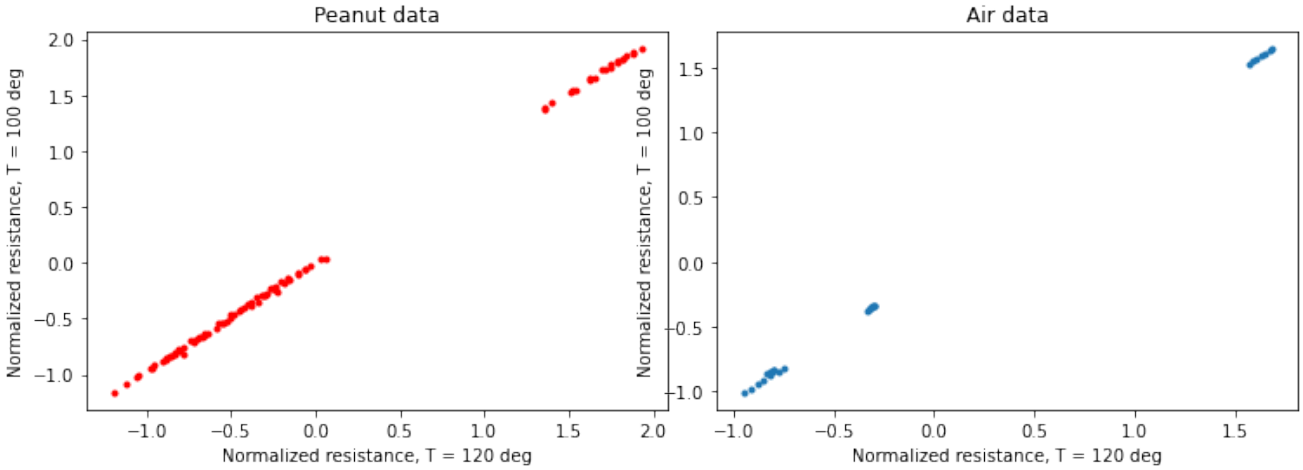
(b) PCA on data from the pure EG sensor.

Figure 16: Score plots from the two sensors at 30% RH

A very important result to note from this figure is that for the sensor with Pt NPs, air is distinguishable from peanut exposures. No such zones can be seen in the score plot of the pure EG sensor. The different peanut weights are overlapping a lot in the score plot from the sensor with Pt NPs. This makes it impossible to determine what peanut weight the sensor is exposed to. On the contrary, the sensor without NPs seem to form clusters for different peanut weights.

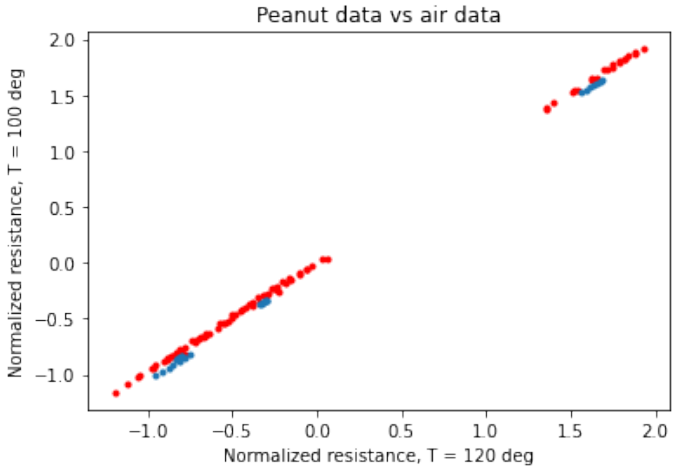
7.2 Data classification

The plot of the normalized reference resistance collected from sensors can be viewed in Figure 17.



(a) Normalized resistance for peanut exposure at 120 vs 100 degrees.

(b) Normalized resistance for air exposure at 120 vs 100 degrees.



(c) Peanut and air exposure at 120 vs 120 degrees in relation to each other.

Figure 17: Graphs showing the normalized resistance for peanut exposure at 120 vs 100 degrees Celsius (a), normalized resistance for air exposure at 120 vs 100 degrees Celsius (b) and air exposure at 120 vs 120 degrees Celsius in relation to each other (c).

The application of a decision tree classifier model to this data, after finding the mean and standard deviation values of the resistances over the four temperatures, resulted in the decision tree, that can be seen in Figure 18. Plotting this as a 1D "decision surface" results in Figure 19.

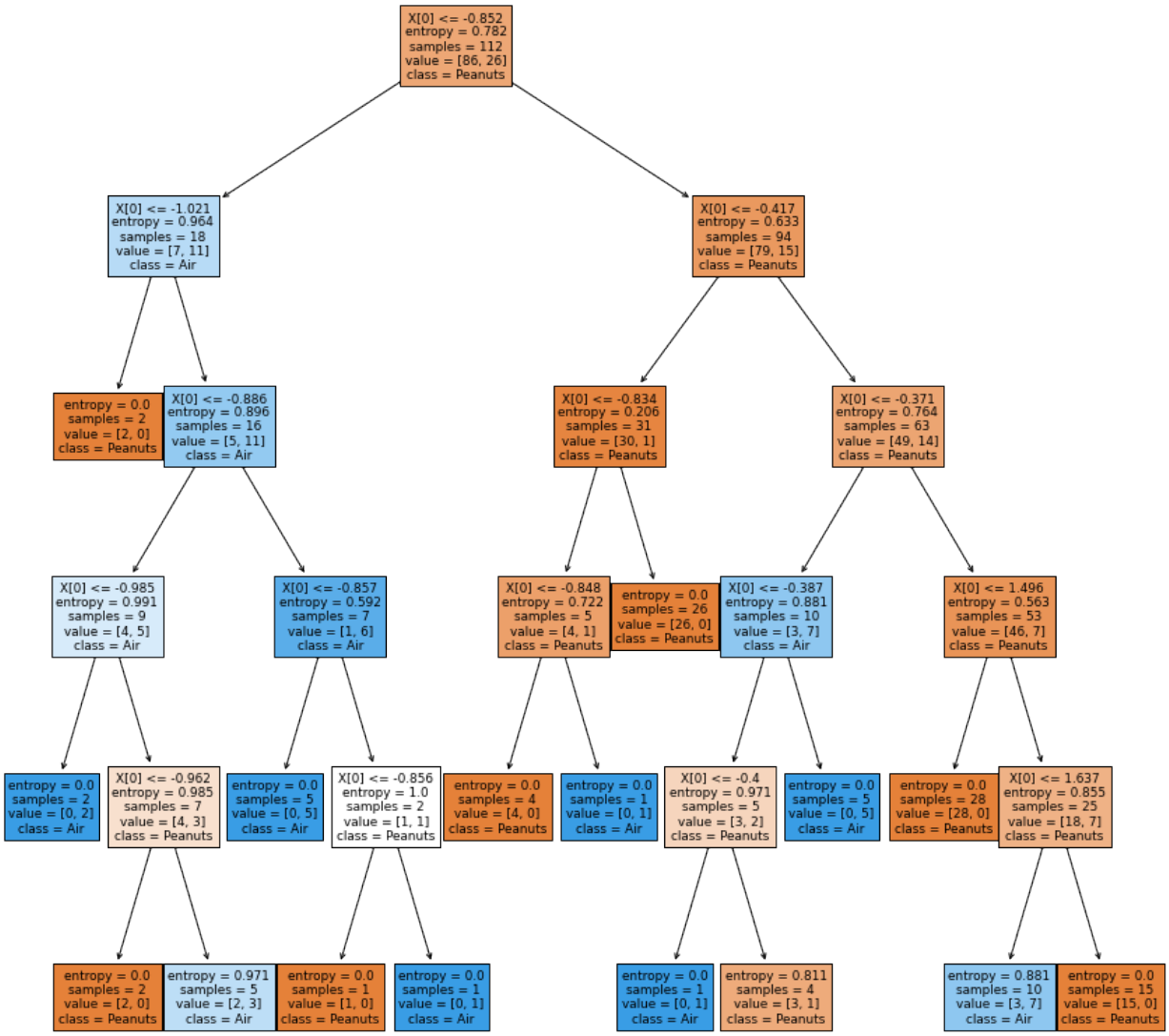


Figure 18: Generated decision tree at 30 % RH and the temperatures 100, 120, 140 and 160 degrees. Blue corresponds to air and orange/red corresponds to peanuts

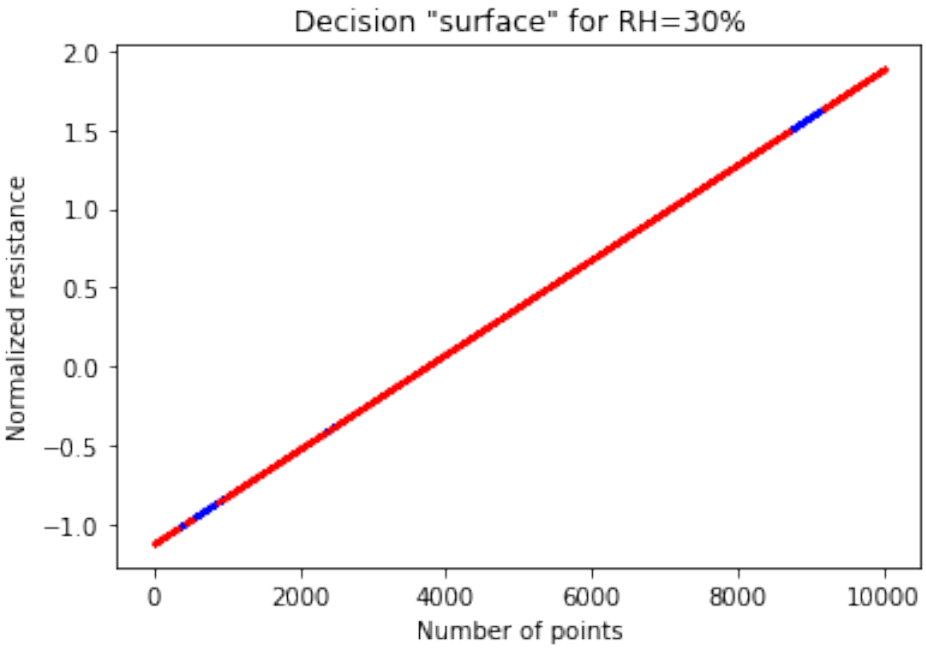


Figure 19: "Decision surface" at 30 % RH and the temperatures 100, 120, 140 and 160 degrees. Blue corresponds to air and orange/red corresponds to peanuts.

The training of the reference data resulted in the following anticipated (theoretical) intervals for normalized peanut- and air resistance can be seen in Table 3.

Normalized Resistance interval (R_{norm})	**Air / peanut interval**
$R_{norm} \leq -0.86$	Peanut
$-0.86 < R_{norm} \leq -0.85$	Air
$-0.85 < R_{norm} \leq 1.5$	Peanut
$1.5 < R_{norm} \leq 1.64$	Air
$R_{norm} > 1.64$	Peanut

Table 3: Peanut- and air interval according to training of the reference data.

7.3 Laboratory setup of hybrid system

The (experimental) intervals for normalized peanut- and air resistance can be viewed in Table 4. By controlling the speaker through arduino code, it makes sounds as long as the normalized resistance value is larger than 2 ohm.

Normalized Resistance interval (R_{norm})	**Air / peanut interval**
$R_{norm} < 1.50$	Peanut
$1.50 \leq R_{norm} < 2.0$	Air
$R_{norm} > 2.0$	Peanut

Table 4: Peanut- and air interval according to laboratory hybrid system setup.

The experimentally obtained values result in the decision tree that can be seen in Figure 20 and corresponding "decision surface" that can be see in Figure 21. The reason for having different decision tree values for testing and validation is explained in the discussion section (section 8).

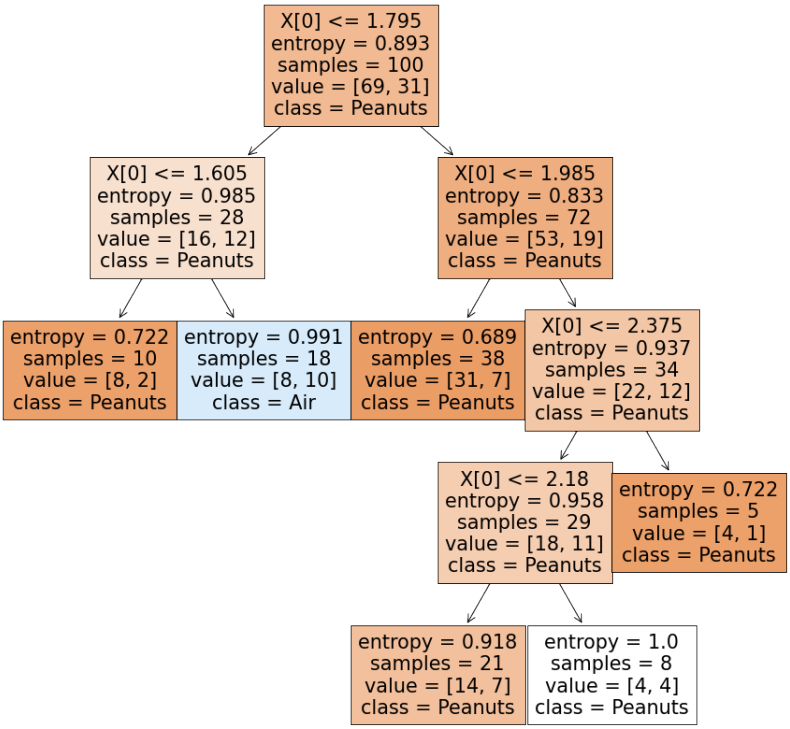


Figure 20: Experimental decision tree at 30 % RH and the temperatures 100, 120, 140 and 160 degrees Celsius. Blue corresponds to air and orange/red corresponds to peanuts.

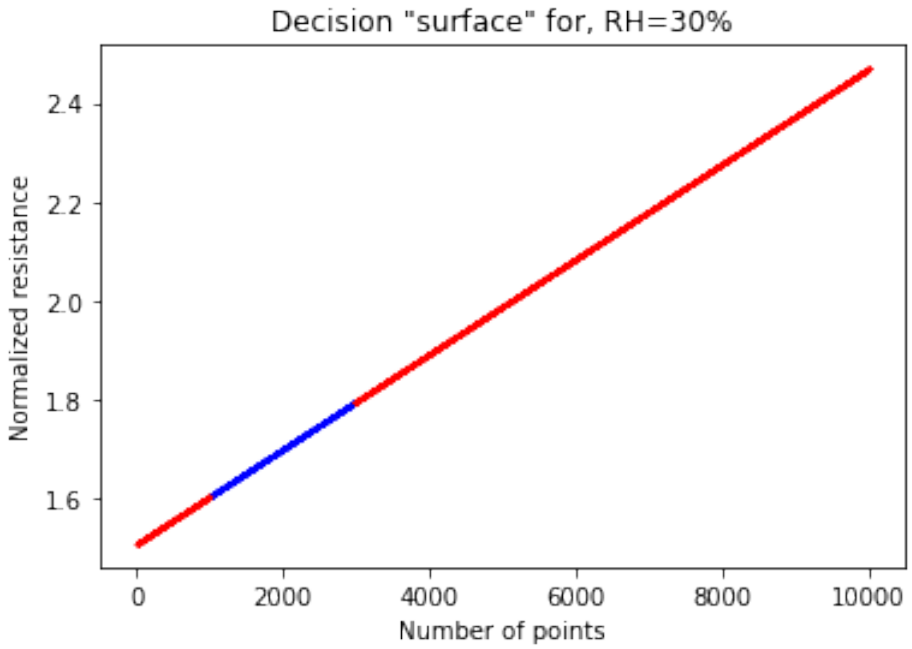


Figure 21: Experimental "decision surface" at 30 % RH and the temperatures 100, 120, 140 and 160 degrees Celsius. Blue corresponds to air and orange/red corresponds to peanuts.

7.4 Demonstrator hybrid system

In Figure 22, the hybrid peanut sensor system built within the limited time of the CDIO project course and exceptional situation of this year's restrictions due to covid-19 is displayed. This system relies on the relative humidity controller and temperature controller from the laboratory. The system is able to distinguish between peanut and air and triggers an alarm when peanut is detected.

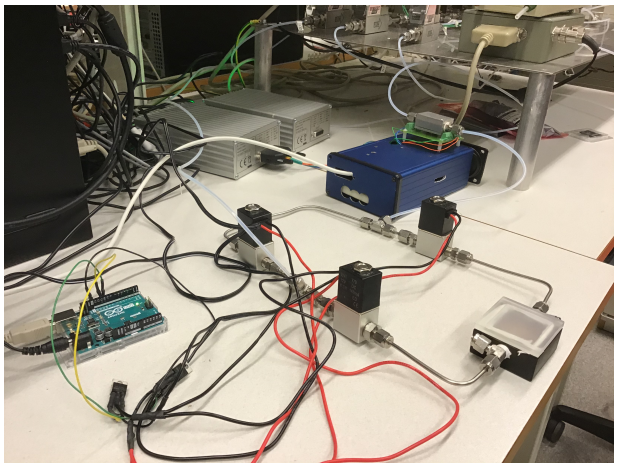


Figure 22: The hybrid peanut sensor system.

8 DISCUSSION

In this section, the different parts of the project and the resulting system are discussed.

8.1 System sensitivity and stability

Through laboratory measurements, the system was observed being very sensitive to regular relative humidity changes, especially when placed in smaller rooms.

The measurements carried out at home were not usable in order to construct the demonstrator, due to fluctuations in the heater and with that, fluctuations in the sensor response. The fluctuations in the heater can be related to the strength of the power source. When too much was connected to the arduino board, the power to the heater became too small to keep in stable. Instead of using the measurements taken at home, reference values taken in the laboratory were used for data analysis and arduino-code construction.

At the beginning of the project, three sensors were constructed for the project group. One of these sensors had an accident (graphene sensor with gold NPs) and with that, it could not be used. From that point only two of the three sensors (one with only graphene and one graphene with platinum NPs) were used for measurements. Out of the two, the one with platinum NPs proved to be the sensor that specifically reacted the most to peanut aroma.

For all measurements, the same peanuts were used. One thought is that, if new ones had been peeled and used for each measurement, the response might have been stronger. If the peanuts go stale over time, this might affect aroma and the emitting gases as well. In addition, over time, the smell disappears as well, which might lead to a weaker response the more times the same peanuts are measured with replacement of fresh ones.

The sensor signal was hard to keep stable and it was hard to get it to stabilise. Even though temperature cycles were used, the sensors needed to stabilise for approximately an hour in order to respond well to a series of alter-

native peanut- and air exposures. This in turn leads to differences between the decision tree generated from the reference data and the decision tree generated from the hybrid system measurement data.

8.2 Data analysis

We saw that the clusters of air measurements were clearly separated from the measurements of peanuts for the sensor decorated with Pt NPs. This suggests that a classifier could be used and integrated in our demonstrator to correctly sound the alarm when exposed to peanuts. Using our method of data classification this was confirmed to be the case. In our demonstrator we are only using the Pt NPs sensor and it can not distinguish between different peanut weights. However, as seen in Figure 16 the peanut weights are somewhat separated for the pure EG sensor. Perhaps a combination of the two sensors could make our demonstrator distinguish between peanut weights. These results are obtained at a fix relative humidity. To expand our demonstrators potential maybe we could make use of the fact that the Pt decorated sensor actually can distinguish between relative humidities.

In the data classification, a decision tree classification model was used. The reason for this choice is related to the results from the PCA and the limitations of the arduino/microcontroller. Through the PCA, it was possible to see that the system could distinguish between peanut and air. Therefore, the algorithm for classification did not need to be more complex than true/not true or peanut/not peanut. With this, the decision tree classification model was optimal.

In the training data, which was used in order to train the model, there were four different temperatures. The limitation of the arduino was that within the time frame of this project, it was not possible to measure the temperature through the arduino board. Therefore, it was unable to know at which part of the temperature cycle the measurement was taken (especially outside of the laboratory). In order to compensate for this limitation, the data needed to be scaled down to one dimension. With this down-scaling, the classifications model that could be applied became limited and the decision tree resulted in the model that generated the most applicable result in terms of interpretation and implementation.

8.3 Implementation

Three different languages/environments were used throughout the project: matlab, python and arduino software. Matlab was a good program to use for the PCA for several reasons. Matlab already had a PCA in-built and it had been used during the Assignments earlier in the project. For the data classification process, python was used. There were several reasons for this choice of language: python has several libraries for different models, which made it easy to implement and compare different ones before deciding on which one to use for the arduino implementation. The second reason for using python was that it could be done though Google colaboratory and with that, all code was accessible online. A third reason was related to the fact that the computer used for the data classification implementation was unable to handle a heavy program as matlab. The arduino software is the software environment directly connected to and used for programming arduino board. In order to get an as smooth interface as possible, the project group deemed it most fitting to use this environment for this purpose.

8.4 Order deliveries

Throughout the project, one limitation was the amount of time for the raw peanuts to be delivered. The plan was to use raw peanuts in order to take measurements on an as natural peanut as possible. Though they were hard to find in stores and supermarkets, it was possible to order them online. Due to the order delivery being delayed, there was no time to retake the measurement once the order arrived. Therefore, the final system is calibrated by using roasted peanuts, which were available in the stores of Linköping. Since, as seen in Table 1, raw and roasted peanuts emits different aroma compounds, it would have been interesting to see whether the system would have reacted differently to raw peanuts compared to roasted ones. Since roasted peanuts are the most commonly used and therefore most likely to be exposed to, the experiments are relevant.

9 FUTURE DEVELOPMENT

- Making a GCMS-measurement to make the device able to distinguish between different gases emitted from the peanuts. This would, in turn, make the fingerprint more exact.
- Using completely raw peanuts, instead of roasted.
- Making sure the device works outside of the laboratory, in a regular indoor environment, by solving the problem concerning the heater stability.
- In some way being able to measure the temperature from the arduino in order to make a more accurate comparison with the reference values.
- The designed box can be made smaller, improved and used in the future.

10 CONCLUSION

In conclusion, it is now known that at least the hybrid system is able to distinguish between peanut and air. Also, from the experimental results in PCA, it has been confirmed that the NPs do improve the sensing for peanut since a clearer pattern of clusters can be seen. Last but not least, it is highly possible for this product to be a portable device if only it is possible to provide a stable voltage to manipulate the temperature for the heater to form a nice temperature cycle.

Appendices

In this section, appendices of data- and code files can be found.

A PCA-CODE, MATLAB

```
%% Load data
clear all
close all
clc

file05g=load('point5g.mat');
file2g=load('2g.mat');
file5g=load('5gnew.mat');
file10g=load('10g.mat');
data05g=file05g.dataPoint5g;
data2g=file2g.data2g;
data5g=file5g.data5g;
data10g=file10g.data10g;

resistance_05g = data05g.Resistance;
resistance_2g = data2g.Resistance;
resistance_5g = data5g.Resistance;
resistance_10g = data10g.Resistance;
%--------------------------
Tempcyc = data5g(data5g.Time < 164,:);
x = 234;

Virtualsensor1_05g = [];
Virtualsensor2_05g = [];
Virtualsensor3_05g = [];
Virtualsensor4_05g = [];

Virtualsensor1_2g = [];
Virtualsensor2_2g = [];
Virtualsensor3_2g = [];
Virtualsensor4_2g = [];

Virtualsensor1_5g = [];
Virtualsensor2_5g = [];
Virtualsensor3_5g = [];
Virtualsensor4_5g = [];
```

```
Virtualsensor1_10g = [];
Virtualsensor2_10g = [];
Virtualsensor3_10g = [];
Virtualsensor4_10g = [];

for N = 1:x
    Virtualsensor1_05g = [Virtualsensor1_05g; data05g(40 +(N-1)*160,:)]; %100
    Virtualsensor2_05g = [Virtualsensor2_05g; data05g(80 +(N-1)*160,:)]; %120
    Virtualsensor3_05g = [Virtualsensor3_05g; data05g(120 +(N-1)*160,:)]; %140
    Virtualsensor4_05g = [Virtualsensor4_05g; data05g(160 +(N-1)*160,:)]; %160

    Virtualsensor1_2g = [Virtualsensor1_2g; data2g(40 +(N-1)*160,:)]; %100
    Virtualsensor2_2g = [Virtualsensor2_2g; data2g(80 +(N-1)*160,:)]; %120
    Virtualsensor3_2g = [Virtualsensor3_2g; data2g(120 +(N-1)*160,:)]; %140
    Virtualsensor4_2g = [Virtualsensor4_2g; data2g(160 +(N-1)*160,:)]; %160

    Virtualsensor1_5g = [Virtualsensor1_5g; data5g(40 +(N-1)*160,:)]; %100
    Virtualsensor2_5g = [Virtualsensor2_5g; data5g(80 +(N-1)*160,:)]; %120
    Virtualsensor3_5g = [Virtualsensor3_5g; data5g(120 +(N-1)*160,:)]; %140
    Virtualsensor4_5g = [Virtualsensor4_5g; data5g(160 +(N-1)*160,:)]; %160

    Virtualsensor1_10g = [Virtualsensor1_10g; data10g(40 +(N-1)*160,:)]; %100
    Virtualsensor2_10g = [Virtualsensor2_10g; data10g(80 +(N-1)*160,:)]; %120
    Virtualsensor3_10g = [Virtualsensor3_10g; data10g(120 +(N-1)*160,:)]; %140
    Virtualsensor4_10g = [Virtualsensor4_10g; data10g(160 +(N-1)*160,:)]; %160
    disp(N)
end

for i=1:x
    mv(i,1)=mean(resistance_05g((35:40)+160*(i-1),1));
    mv(i,2)=mean(resistance_05g((75:80)+160*(i-1),1));
    mv(i,3)=mean(resistance_05g((115:120)+160*(i-1),1));
    mv(i,4)=mean(resistance_05g((155:160)+160*(i-1),1));
end

for i=1:x
    mv(i+x,1)=mean(resistance_2g((35:40)+160*(i-1),1));
    mv(i+x,2)=mean(resistance_2g((75:80)+160*(i-1),1));
    mv(i+x,3)=mean(resistance_2g((115:120)+160*(i-1),1));
    mv(i+x,4)=mean(resistance_2g((155:160)+160*(i-1),1));
end

for i=1:x
    mv(i+(2*x),1)=mean(resistance_5g((35:40)+160*(i-1),1));
    mv(i+(2*x),2)=mean(resistance_5g((75:80)+160*(i-1),1));
```

```
    mv(i+(2*x),3)=mean(resistance_5g((115:120)+160*(i-1),1));
    mv(i+(2*x),4)=mean(resistance_5g((155:160)+160*(i-1),1));
end
for i=1:x
    mv(i+(3*x),1)=mean(resistance_10g((35:40)+160*(i-1),1));
    mv(i+(3*x),2)=mean(resistance_10g((75:80)+160*(i-1),1));
    mv(i+(3*x),3)=mean(resistance_10g((115:120)+160*(i-1),1));
    mv(i+(3*x),4)=mean(resistance_10g((155:160)+160*(i-1),1));
end
%%
for i=1:4 %LOOP 4 TIMES FOR THE 4 TEMPERATURES
    COI((1:7),i)=mv((103:109),i);%EXPOSURE 1 0.5g 30%RH
    COI((8:14),i)=mv((121:127),i);%EXPOSURE 2 0.5g 30%RH
    COI((15:21),i)=mv((139:145),i);%EXPOSURE 3 0.5g 30%RH

    COI((22:28),i)=mv((x+103:x+109),i);%EXPOSURE 1 2g 30%RH
    COI((29:35),i)=mv((x+121:x+127),i);%EXPOSURE 2 2g 30%RH
    COI((36:42),i)=mv((x+139:x+145),i);%EXPOSURE 3 2g 30%RH

    COI((43:49),i)=mv((3*x+103:3*x+109),i);%EXPOSURE 1 10g 30%RH
    COI((50:56),i)=mv((3*x+121:3*x+127),i);%EXPOSURE 2 10g 30%RH
    COI((57:63),i)=mv((3*x+139:3*x+145),i);%EXPOSURE 3 10g 30%RH

    COI((64:70),i)=mv((94:100),i);%AIR from 0.5g 30%
    COI((71:77),i)=mv((x+94:x+100),i);%AIR from 2g 30%
    COI((78:84),i)=mv(((x*3)+94:(x*3)+100),i);%AIR from 10g 30%
end
COI_mean=mean(COI);
COI_std=std(COI);
COI_norm=((COI-COI_mean)./COI_std);

% Define colors for PCA
c1=[0.2 0 0]; %dark red - 5%RH 0.5g
c2=[0.5 0 0]; %dark red - 5%RH 2g
c3=[0.7 0 0]; %dark red - 5%RH 5g
c4=[1 0 0]; %red - 5%RH 10g

c5=[0 0.2 0]; %dark green - 30%RH 0.5g
c6=[0 0.5 0]; %dark green - 30%RH 2g
c7=[0 0.7 0]; %dark green - 30%RH 5g
c8=[0 1 0]; %green- 30%RH 10g

c9=[0 0 0.2]; %dark blue - 60%RH 0.5g
c10=[0 0 0.5]; %dark blue - 60%RH 2g
c11=[0 0 0.7]; %dark blue - 60%RH 5g
```

```
c12=[0 0 1]; %blue - 60%RH 10g

c13=[1 0.1 0.6]; %AIR

for i=1:7
    colors{i}=c2;
    colors{i+7}=c2;
    colors{i+14}=c2;

    colors{i+21}=c2;
    colors{i+28}=c2;
    colors{i+35}=c2;

    colors{i+42}=c2;
    colors{i+49}=c2;
    colors{i+56}=c2;

    colors{i+63}=c11;
    colors{i+70}=c11;
    colors{i+77}=c11;
end

[coeff,score,latent,tsq,exp]=pca(COI_norm(:,:));

figure(1)
plot(Tempcyc.Time, Tempcyc.Resistance)
figure(2)
plot(data5g.Time, data5g.Resistance)
figure(3)
plot(Virtualsensor1_5g.Time, Virtualsensor1_5g.Resistance)
title('100 C ')
figure(4)
plot(Virtualsensor2_5g.Time, Virtualsensor2_5g.Resistance)
title('120 C ')
figure(5)
plot(Virtualsensor2_5g.Time, Virtualsensor3_5g.Resistance)
title('140 C ')
figure(6)
plot(Virtualsensor2_5g.Time, Virtualsensor4_5g.Resistance)
title('160 C ')
figure(7)
plot(mv(:,2))

figure(8)
for i=1:84
```

```
  hold on;
  plot(score(i,1),score(i,2),'^','markersize',4,'linewidth',2,'color',colors{i});
end
explain =(latent/sum(latent))*100;
xlabel(strcat('PC1 (%)= ',num2str(explain(1))));
ylabel(strcat('PC2 (%)= ',num2str(explain(2))));
figure(9)
vbls = {'100','120','140','160'}; % Labels for the variables
biplot(coeff(:,1:3),'Scores',score(:,1:3),'VarLabels',vbls);
```

B PYTHON CODE FOR REFERENCE DATA CLASSIFICATION

```
# -*- coding: utf-8 -*-
"""Untitled0.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/19ALUxeMZYPgCmxqQlC31xj9IZQ5YcbrV
"""

print(__doc__)

#Import of libraries and model
import numpy as np
import matplotlib.pyplot as plt
import scipy.io
import numpy as np
import csv

from sklearn.tree import DecisionTreeClassifier, plot_tree


# Parameters definition
n_classes = 2
plot_colors = "rb"
plot_step = 0.02

#Loads the matlab-files into the Python environment
Coeff = scipy.io.loadmat('30RH_COEFF.mat')
score_stable_RH = scipy.io.loadmat('30RH_SCORE.mat')
COI_MEAN = scipy.io.loadmat('30RH_COI_MEAN')
COI = scipy.io.loadmat('30RH_COI.mat')

#Gives us a shape of the data content that we are interested in
Coeff['coeff'].shape
COI['COI'].shape
score_stable_RH['score'].shape
COI_MEAN['COI_mean'].shape

#Extracts the correct data from the file
score_stable_RH = score_stable_RH['score']
coeff=Coeff['coeff']
coi= COI['COI']
```

```
#Doublechecks that the mean of coeff is the same as the first column of the COI-matrix
coeff.mean(axis=0)
coeff_mean_axis0= coi.mean(axis=0)
coeff_mean_axis1 = np.mean(coeff_mean_axis0)
COI_MEAN

#Doublechecks that the standard deviation of coeff is the same as the first column of
#the COI-matrix
coi.std(axis=0)
coi_std_mean = np.mean(coi.std(axis=0))
coi_mean_mean = np.mean(coi.mean(axis=0))

#Finds the normalised values of COI
coi_norm = (coi-coi.mean(axis=0))/coi.std(axis=0)

#Finds the mean of the normalised values of COI
coi_norm_mean = np.mean(coi_norm, axis=1)

#Finds length of array
array_length = len(coi_norm_mean)

# Turns array into matrix and transponates it
coi_norm_mean_matrix = np.asmatrix(coi_norm_mean)
transponate_coi_norm_mean_matrix = coi_norm_mean_matrix.transpose()

#Doublechecks that the mean and STD of COI mean
coi_norm.mean(axis=0)
coi_norm_mean_mean = np.mean(coi_norm.mean(axis=0))
coi_norm.std(axis=0)
coi_norm_std_mean = np.mean(coi_norm.std(axis=0))

#Definition of x-axis (X) and y-axis (y) for model training
X=transponate_coi_norm_mean_matrix
X.shape
y = np.zeros(len(transponate_coi_norm_mean_matrix))
y[86:] = 1
labels = ['Peanuts','Air']

#Model code that trains the data we send into it.
clf = DecisionTreeClassifier(criterion='entropy',max_depth=5).fit(X, y)


# Plot the decision boundary
plt.figure(figsize=(10,10))
```

```
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1

#Plots the decision tree
plt.figure(figsize=(15,15))
#clf = DecisionTreeClassifier().fit(X,y)
plot_tree(clf, filled=True, class_names=labels)
plt.show()

#Plots corresponding decision "surface"
X_test = np.linspace(X.min(), X.max(), 10000)
X_test = X_test.reshape(-1,1)
y_test = clf.predict(X_test)

plt.plot(X_x[y_test==1], X_test[y_test==1],'.', markersize=1, color='blue')
plt.plot(X_x[y_test==0], X_test[y_test==0],'.', markersize=1, color='red')
plt.title('Decision "surface" for, RH=30%')
plt.xlabel('Number of points')
plt.ylabel('Normalized resistance')
```

C PYTHON CODE FOR HYBRID SYSTEM DATA CLASSIFICATION

```
# -*- coding: utf-8 -*-
"""Untitled0.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/19ALUxeMZYPgCmxqQlC31xj9IZQ5YcbrV
"""

#Import of libraries and models
print(__doc__)

import numpy as np
import matplotlib.pyplot as plt
import scipy.io
import numpy as np
import csv

from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier, plot_tree

#Parameter definitions
# Parameters
n_classes = 2
plot_colors = "rb"
plot_step = 0.02

#Opens the uploaded file with hybrid system measurement data.
serial = open('serialcopy.txt', 'r')
serial.read()

#Matrix from all measurements.
serial_matrix_all = np.array([[-0.21], [0.17],[0.17],[-0.40],[-0.78],[-1.16],[-1.54],[-1.54],[-1.54],
[-1.54],[-1.35],[-1.16],[-1.16],[-0.59],[-0.02],[0.36],[1.13],[1.32],[1.51],[1.70],[1.70],[1.89],
[1.89],[1.89],[1.70],[1.51],[1.32],[1.13],[1.13],[1.13],[1.32],[1.32],[1.32],[1.32],[1.13],[1.32],
[1.51],[1.51],[1.89],[1.89],[1.89],[1.70],[1.51],[1.51],[1.51],[1.70],[1.70],[1.51],[1.51],[1.51],
[1.51],[1.89],[1.89],[1.89],[1.89],[1.89],[1.89],[2.08],[2.28],[2.08],[2.08],[2.08],[2.08],[2.28],
[2.47],[2.47],[2.47],[2.47],[2.28],[2.28],[2.08],[1.89],[1.89],[1.89],[1.89],[2.08],[1.89],[1.89],
[1.70],[1.89],[1.89],[1.89],[1.89],[1.70],[1.89],[1.89],[1.89],[1.89],[1.70],[1.70],[1.89],[1.89],
[2.08],[2.08],[1.89],[1.89],[1.70],[1.51],[1.51],[1.51],[1.51],[1.70],[1.70],[1.70],[1.70],[1.70],
[1.51],[1.51],[1.51],[1.70],[1.89],[2.08],[2.08],[1.89],[2.08],[1.89],[2.08],[2.08],[1.89],[1.89],
[1.70],[1.89],[1.89],[1.89],[2.08],[1.89],[1.89],[1.89],[1.89],[2.08],[2.08],[2.28],[2.08],[2.08],
```

```
[2.28],[2.28],[2.47],[2.28],[2.08],[2.08],[1.89],[1.89],[1.70],[1.70],[1.70],[1.70],[1.70],[1.70]])
serial_matrix_all.shape

#Separation of peanut exposure values (peanut_1, peanut_2) and air exposure values (air_1, air_2)
peanut_1 = np.array([[1.51],[1.51],[1.51],[1.89],[1.89],[1.89],[1.89],[1.89],[1.89],[2.08],[2.28],
[2.08],[2.08],[2.08],[2.08]])
peanut_1.shape

air_1 = np.array([[2.28],[2.47],[2.47],[2.47],[2.47],[2.28],[2.28],[2.08],[1.89],[1.89],[1.89],
[1.89],[2.08],[1.89],[1.89],[1.70],[1.89],[1.89],[1.89],[1.89],[1.70],[1.89],[1.89],[1.89],[1.89],
[1.70],[1.70],[1.89],[1.89],[2.08],[2.08],[1.89],[1.89],[1.70],[1.51],[1.51],[1.51],[1.51],[1.70],
[1.70],[1.70],[1.70],[1.70],[1.51],[1.51]])
air_1.shape

peanut_2 = np.array([[1.51],[1.70],[1.89],[2.08],[2.08],[1.89],[2.08],[1.89],[2.08],[2.08],[1.89],
[1.89],[1.70],[1.89],[1.89]])
peanut_2.shape

air_2 = np.array([[1.89],[2.08],[1.89],[1.89],[1.89],[1.89],[2.08],[2.08],[2.28],[2.08],[2.08],
[2.28],[2.28],[2.47],[2.28],[2.08],[2.08],[1.89],[1.89],[1.70],[1.70],[1.70],[1.70],[1.70],[1.70]])
air_2.shape

# Peanut and air exposure values reorganized into a matrix where all
#air values are last in the matrix.
peanut_air_matrix = np.array([[1.51],[1.51],[1.51],[1.89],[1.89],[1.89],[1.89],[1.89],[1.89],[2.08],
[2.28],[2.08],[2.08],[2.08],[2.08],[1.51],[1.70],[1.89],[2.08],[2.08],[1.89],[2.08],[1.89],[2.08],
[2.08],[1.89],[1.89],[1.70],[1.89],[1.89],[2.28],[2.47],[2.47],[2.47],[2.47],[2.28],[2.28],[2.08],
[1.89],[1.89],[1.89],[1.89],[2.08],[1.89],[1.89],[1.70],[1.89],[1.89],[1.89],[1.89],[1.70],[1.89],
[1.89],[1.89],[1.89],[1.70],[1.70],[1.89],[1.89],[2.08],[2.08],[1.89],[1.89],[1.70],[1.51],[1.51],
[1.51],[1.51],[1.70],[1.70],[1.70],[1.70],[1.70],[1.51],[1.51],[1.89],[2.08],[1.89],[1.89],[1.89],
[1.89],[2.08],[2.08],[2.28],[2.08],[2.08],[2.28],[2.28],[2.47],[2.28],[2.08],[2.08],[1.89],[1.89],
[1.70],[1.70],[1.70],[1.70],[1.70],[1.70]])
peanut_air_matrix.shape

#Definition of x-axis (X) and y-axis (y) for model training
X=peanut_air_matrix
X.shape
y = np.zeros(len(peanut_air_matrix))
y[69:] = 1
labels = ['Peanuts','Air']

#Model code that trains the data we send into it.
clf = DecisionTreeClassifier(criterion='entropy',max_depth=1000).fit(X, y)

# Plot the decision boundary
```

```
plt.figure(figsize=(10,10))

x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1

#Plots the decision tree
plt.figure(figsize=(15,15))
plot_tree(clf, filled=True, class_names=labels)
plt.show()

X_test = np.linspace(X.min(), X.max(), 10000)
X_test = X_test.reshape(-1,1)
y_test = clf.predict(X_test)

#Plots corresponding decision "surface"
plt.plot(X_x[y_test==1], X_test[y_test==1],'.', markersize=1, color='blue')
plt.plot(X_x[y_test==0], X_test[y_test==0],'.', markersize=1, color='red')
plt.title('Decision "surface" for, RH=30%')
plt.xlabel('Number of points')
plt.ylabel('Normalized resistance')
```

D ARDUINO CODE FOR RESISTANCE COMPARISON

```
//In this implementation we use a mean value of all temperature cycles,
// therefore we can use the temperature cycles to control the drift of the sensor.

//constants and variabels definitions
const int gas = 0;
int speakPin = 9;
const float mean_value_measurement = 5315.696541966148;
const float std_value_measurement = 43.501715044544255;

int LEDPin = 7;
int peanutPin_value = A0;
int i = 0;

//floats for saving five measured values
float value_1;
float value_2;
float value_3;
float value_4;

void setup(){
  Serial.begin(9600);
  pinMode(LEDPin, OUTPUT);
  digitalWrite(LEDPin, LOW);

delay(35000);
value_1 = analogRead(peanutPin_value);
Serial.println(value_1);
delay(40000);
value_2 = analogRead(peanutPin_value);
Serial.println(value_2);
delay(40000);
value_3 = analogRead(peanutPin_value);
Serial.println(value_3);
delay(40000);
value_4 = analogRead(peanutPin_value);
Serial.println(value_4);

}
int j = 0;

void loop()
{
```

```
//adds one to i and saves value from the most recent reading and replace the
//values from previous readings.
if (j == 0)
{
  j = 1;
}
else{
value_1 = value_2;
value_2 = value_3;
value_3 = value_4;
delay(40000);
value_4 = analogRead(peanutPin_value); //reads the latest value from analog pin and saves it.
}

//Calculates the mean of read sensor values.
float mean_peanutPin_value = (value_1 + value_2 + value_3 + value_4)/4;

// calculates the sensor output in volts.
float mean_peanutSensorValue = (mean_peanutPin_value*5)/1024;
Serial.print(mean_peanutSensorValue);
Serial.print(",");

// transforms the sensor value to resistance through voltage division.
// 1000 is the known resistance from the voltage division
float mean_peanutSensorResistance = ((22000*mean_peanutSensorValue)/(5-mean_peanutSensorValue));
Serial.print(mean_peanutSensorResistance);
Serial.print(",");

//normalises the sensor resistance into a comparable value.
float mean_norm_peanutSensorResistance =
  ((mean_peanutSensorResistance - mean_value_measurement)/std_value_measurement);
 Serial.println(mean_norm_peanutSensorResistance);

// comparison with the values from the decision tree. If we are in a peanut zone,
// the alarm will be triggered through the warningPin.
  if (mean_norm_peanutSensorResistance > -0.852 && mean_norm_peanutSensorResistance <= 1.496)
    {
    digitalWrite(LEDPin, HIGH);
    tone(speakPin, 3000, 50);
    digitalWrite(LEDPin, LOW);
    }
  else if (mean_norm_peanutSensorResistance <= -0.856)
    {
    digitalWrite(LEDPin, HIGH);
    tone(speakPin, 3000, 50);
```

```
    digitalWrite(LEDPin, LOW);
    }
  else if (mean_norm_peanutSensorResistance > 1.637)
    {
    digitalWrite(LEDPin, HIGH);
    tone(speakPin, 3000, 50);
    digitalWrite(LEDPin, LOW);
    }
}
```

E ARDUINO CODE FOR VALVE CONTROL

```
// Variable definitions
int peanutvalvePin = 11;
int airvalvePin = 10;

// Functions that run once. Makes the valves respond to digital signals.
void setup() {
digitalWrite(peanutvalvePin,LOW);
digitalWrite(airvalvePin,LOW);
pinMode(peanutvalvePin, OUTPUT);
pinMode(airvalvePin, OUTPUT);
}

// Loop that controls the opening and closing of the valves.
void loop() {
digitalWrite(airvalvePin,HIGH);
delay(10000);
//First exposure
digitalWrite(airvalvePin,LOW);
digitalWrite(peanutvalvePin,HIGH);
delay(600000);
digitalWrite(peanutvalvePin,LOW);
digitalWrite(airvalvePin,HIGH);
delay(1800000);
}
```

F 30RH_COI.MAT

```
{'COI': array([[5617.75631628, 5405.50243617, 5159.81907804, 4907.65682626],
       [5628.72188161, 5413.92978551, 5165.84518855, 4911.84527939],
       [5634.26463741, 5419.06503557, 5169.87657671, 4914.99941317],
       [5638.01778833, 5422.46564908, 5172.62444771, 4916.72474146],
       [5640.75223788, 5424.84313936, 5174.89284604, 4918.82782639],
       [5642.93968476, 5426.90332962, 5176.80982206, 4920.45296958],
       [5644.0400982 , 5427.47751525, 5176.86794419, 4920.21366747],
       [5621.53191814, 5408.67418588, 5162.77344864, 4911.04140116],
       [5627.50382506, 5413.36822902, 5166.48352346, 4913.84591673],
       [5631.22477253, 5416.94402384, 5169.21843441, 4916.0981266 ],
       [5634.0391414 , 5419.56632399, 5171.26216455, 4917.22172699],
       [5635.92699134, 5421.20002595, 5172.8096306 , 4918.39258935],
       [5637.28120963, 5422.35643614, 5173.96925729, 4919.36872507],
       [5637.43157147, 5421.82748432, 5173.15224873, 4918.58288392],
       [5617.46879356, 5405.33179523, 5160.66553034, 4909.41581339],
       [5622.15876289, 5408.91760141, 5163.16291228, 4911.53017774],
       [5624.82484691, 5411.4688993 , 5165.01940052, 4912.89648948],
       [5626.86829491, 5413.27592749, 5166.57380506, 4914.28599469],
       [5628.37096509, 5414.80579566, 5167.86908919, 4915.39704703],
       [5629.80960575, 5416.00680143, 5169.05879548, 4916.08985433],
       [5629.93795097, 5415.72774347, 5168.3427467 , 4915.30482713],
       [5645.78957655, 5430.52770411, 5180.61564554, 4925.51613603],
       [5654.82293123, 5437.8890959 , 5186.11398405, 4929.61726203],
       [5660.58124041, 5443.00397803, 5190.18855286, 4932.82860936],
       [5664.6049627 , 5446.86158371, 5193.61563795, 4935.63718775],
       [5668.11526864, 5450.0596913 , 5196.40845485, 4938.00814919],
       [5671.07895486, 5452.75722385, 5198.72441247, 4939.87002476],
       [5672.52167206, 5453.17999763, 5198.7337761 , 4939.53234153],
       [5648.7331169 , 5433.20814842, 5183.7079633 , 4929.05203778],
       [5654.61161274, 5438.08224414, 5187.40721371, 4931.81508147],
       [5658.2201733 , 5441.59059231, 5190.09306686, 4934.08886039],
       [5661.39129788, 5444.67134506, 5192.87708652, 4936.33136814],
       [5664.50180477, 5447.32878771, 5195.11217835, 4937.98138355],
       [5666.41749324, 5449.08698045, 5196.72673895, 4939.16888444],
       [5666.86142971, 5448.90235558, 5196.178199  , 4938.40294402],
       [5644.95885335, 5430.20574906, 5181.40722637, 4927.99573806],
       [5650.13523303, 5434.67560368, 5185.0217753 , 4930.74742959],
       [5654.16577712, 5438.36697533, 5187.98081229, 4933.11835799],
       [5657.24005564, 5441.09482166, 5190.44340013, 4934.91583435],
       [5659.70610505, 5443.52317535, 5192.43196622, 4936.72312293],
       [5661.64653279, 5445.36896547, 5194.05775501, 4937.68943405],
       [5662.06010621, 5444.83047171, 5193.13314126, 4936.81032784],
```

```
[5736.65262513, 5513.80233665, 5258.12697375, 4996.45035271],
[5745.84390343, 5521.43085459, 5263.80331682, 5000.75856984],
[5751.74348268, 5526.69932764, 5268.23138604, 5004.24011184],
[5756.26160761, 5530.89188277, 5271.744436  , 5006.98493836],
[5759.8127724 , 5534.21863868, 5274.6229878 , 5009.49940379],
[5762.8568523 , 5536.85959538, 5276.9872407 , 5011.73763396],
[5765.22005063, 5538.22193985, 5277.45698362, 5011.16310372],
[5736.63201016, 5514.06079127, 5259.09185542, 4998.35281132],
[5744.16420533, 5520.61409294, 5264.39834005, 5002.39233696],
[5750.13175553, 5526.0347961 , 5268.78198373, 5005.6797361 ],
[5754.34101162, 5529.82238815, 5271.965134  , 5008.31843604],
[5757.90812192, 5532.98428361, 5274.7122245 , 5010.62422124],
[5760.79852945, 5535.70990756, 5277.09649362, 5012.66137636],
[5762.44600838, 5536.3831943 , 5277.1059293 , 5012.24168984],
[5738.85181785, 5516.41293865, 5261.63757246, 5000.6585421 ],
[5744.90827412, 5521.65293636, 5265.80494418, 5004.08647449],
[5749.82767571, 5526.29203709, 5269.56328102, 5007.00587932],
[5753.69265163, 5529.67934317, 5272.36407218, 5008.97322191],
[5756.08906659, 5531.9095591 , 5274.13975417, 5010.34966474],
[5758.19851689, 5533.79437078, 5275.88136005, 5012.15731072],
[5759.58641439, 5534.45589532, 5276.10236115, 5011.83136146],
[5610.62972017, 5398.7533365 , 5154.27270517, 4903.20833333],
[5625.978841  , 5411.9291995 , 5164.42814117, 4910.95524067],
[5636.72371433, 5421.53995767, 5172.4273275 , 4916.67683917],
[5644.96240233, 5428.85921233, 5178.40006533, 4921.58821617],
[5650.9269205 , 5434.264974  , 5183.24145517, 4925.310791  ],
[5655.45198567, 5438.58365883, 5186.71639017, 4928.953125  ],
[5658.4581705 , 5439.53556317, 5186.33374017, 4926.99218767],
[5614.59358733, 5402.45646167, 5158.50732433, 4907.4592285 ],
[5626.2319335 , 5412.60042317, 5166.55224583, 4913.66487633],
[5635.14933267, 5420.873291  , 5173.46752933, 4919.09049467],
[5642.23453783, 5427.10351567, 5178.45711267, 4923.0058595 ],
[5647.25667317, 5431.688151  , 5182.12622067, 4925.94718433],
[5650.916911  , 5435.045166  , 5185.086507  , 4928.782959  ],
[5652.84366867, 5435.656087  , 5185.389974  , 4929.38053383],
[5622.35074867, 5410.01342783, 5166.11263033, 4915.15934267],
[5633.21752917, 5419.62263983, 5173.73893233, 4920.9047855 ],
[5641.4329425 , 5427.11442067, 5179.69807933, 4925.41878267],
[5647.64322917, 5432.72363267, 5184.4039715 , 4929.19270833],
[5652.62150067, 5437.17830383, 5188.26944983, 4932.18261733],
[5656.618978  , 5440.89119483, 5191.36393217, 4934.99088533],
[5657.92919917, 5440.52921533, 5189.92749033, 4932.74601233],
[5631.22804605, 5414.02864637, 5163.46492035, 4907.46607359],
[5630.90390736, 5413.95859533, 5163.63420962, 4907.49148419],
[5630.21143923, 5413.63111758, 5163.42460779, 4907.40724093],
```

```
       [5630.12540694, 5413.49022814, 5163.24114337, 4907.50879164],
       [5629.59682014, 5413.09852147, 5163.43954144, 4907.68849647],
       [5629.20009578, 5412.76194391, 5163.24118192, 4907.8111859 ],
       [5628.54521875, 5412.25808141, 5162.86183439, 4907.57526999],
       [5654.73507166, 5435.84423715, 5182.11894785, 4924.24365011],
       [5654.62864064, 5435.87430294, 5182.30374271, 4924.77606332],
       [5654.8360819 , 5435.92018706, 5182.37748979, 4924.59178556],
       [5654.04090665, 5435.59944082, 5182.30696198, 4924.74425215],
       [5654.06787783, 5435.36832937, 5182.23658707, 4925.02474476],
       [5653.45016733, 5434.96341023, 5182.26651479, 4924.92335021],
       [5652.99826596, 5434.5441747 , 5181.80436691, 4924.69104094],
       [5752.91589039, 5525.9508812 , 5265.77304547, 5000.72858888],
       [5752.36428343, 5525.256184  , 5265.27870395, 5000.49765185],
       [5751.30973387, 5524.41865824, 5264.696078  , 5000.0641981 ],
       [5750.38694619, 5523.55614767, 5264.10553684, 4999.74417152],
       [5749.2337192 , 5522.71503486, 5263.50086499, 4999.2392059 ],
       [5748.22607164, 5521.75890154, 5262.7069124 , 4998.79098087],
       [5747.12789594, 5520.79474329, 5262.00830792, 4998.26453263],
       [5632.53141283, 5414.52189117, 5163.07316083, 4906.4847005 ],
       [5631.17952483, 5412.95434567, 5162.0894365 , 4905.56575533],
       [5629.40747083, 5411.70060233, 5160.8879395 , 4904.794515  ],
       [5627.5146485 , 5409.86930317, 5159.67683917, 4904.11979167],
       [5626.3108725 , 5408.8939615 , 5158.9542645 , 4903.56958   ],
       [5624.40193683, 5407.17032883, 5157.74869783, 4902.59944667],
       [5622.6816405 , 5405.76399733, 5156.47648117, 4901.74104817]]),
'__globals__': [],
'__header__': b'MATLAB 5.0 MAT-file, Platform: PCWIN64, Created on: Sun Dec 13 16:24:45 2020',
'__version__': '1.0'}
```

G 30RH_COI_MEAN.MAT

{'COI_mean': array([[5669.67263719, 5451.43243566, 5199.5742326 , 4942.10686241]]),
 '__globals__': [],
 '__header__': b'MATLAB 5.0 MAT-file , Platform: PCWIN64, Created on: Fri Dec 11 14:01:34 2020',
 '__version__': '1.0'}

H 30RH_COI_STD.MAT


```
{'COI_std': array([[49.68309427, 45.59953177, 41.64720563, 37.85908574]]),
 '__globals__': [],
 '__header__': b'MATLAB 5.0 MAT-file , Platform: PCWIN64, Created on: Fri Dec 11 14:01:57 2020',
 '__version__': '1.0'}
```

I SERIALCOPY.TXT

```
// Initial Arduino
//values

194.00
187.00
211.00
204.00

// output voltage(V),
//resistance(ohm),
//norm. resistance(ohm)

0.97,5306.67,-0.21
0.97,5323.23,0.17
0.97,5323.23,0.17
0.97,5323.23,0.17
0.97,5298.39,-0.40
0.97,5281.86,-0.78
0.97,5265.36,-1.16
0.96,5248.87,-1.54
0.96,5248.87,-1.54
0.96,5248.87,-1.54
0.96,5248.87,-1.54
0.96,5257.11,-1.35
0.97,5265.36,-1.16
0.97,5265.36,-1.16
0.97,5290.13,-0.59
0.97,5314.94,-0.02
0.98,5331.51,0.36
0.98,5364.71,1.13
0.98,5373.03,1.32
0.98,5381.34,1.51
0.98,5389.67,1.70
0.98,5389.67,1.70
0.99,5397.99,1.89
0.99,5397.99,1.89
0.99,5397.99,1.89
0.98,5389.67,1.70
0.98,5381.34,1.51
0.98,5373.03,1.32
0.98,5364.71,1.13

0.98,5364.71,1.13
0.98,5364.71,1.13
0.98,5373.03,1.32
0.98,5373.03,1.32
0.98,5373.03,1.32
0.98,5364.71,1.13
0.98,5373.03,1.32
0.98,5381.34,1.51
0.98,5381.34,1.51
0.99,5397.99,1.89
0.99,5397.99,1.89
0.99,5397.99,1.89
0.98,5389.67,1.70
0.98,5381.34,1.51
0.98,5381.34,1.51
0.98,5381.34,1.51
0.98,5389.67,1.70
0.98,5389.67,1.70
0.98,5381.34,1.51
0.98,5381.34,1.51
0.98,5381.34,1.51
0.99,5397.99,1.89
0.99,5406.33,2.08
0.99,5414.66,2.28
0.99,5406.33,2.08
0.99,5406.33,2.08
0.99,5414.66,2.28
0.99,5423.01,2.47
0.99,5423.01,2.47
0.99,5423.01,2.47
0.99,5414.66,2.28
0.99,5414.66,2.28

0.99,5406.33,2.08
0.99,5397.99,1.89
0.99,5397.99,1.89
0.99,5406.33,2.08
0.99,5406.33,2.08
0.99,5397.99,1.89
0.99,5397.99,1.89
0.99,5397.99,1.89
0.98,5389.67,1.70
0.99,5397.99,1.89
0.99,5397.99,1.89
0.99,5397.99,1.89
0.99,5397.99,1.89
0.99,5397.99,1.89
0.99,5397.99,1.89
0.98,5389.67,1.70
0.98,5381.34,1.51
0.98,5381.34,1.51
0.98,5381.34,1.51
0.98,5389.67,1.70
0.98,5389.67,1.70
0.98,5389.67,1.70
0.98,5389.67,1.70
0.98,5389.67,1.70
0.98,5381.34,1.51
0.98,5381.34,1.51
0.98,5389.67,1.70
0.99,5397.99,1.89

0.99,5406.33,2.08
0.99,5406.33,2.08
0.99,5397.99,1.89
0.99,5406.33,2.08
0.99,5397.99,1.89
0.99,5406.33,2.08
0.99,5406.33,2.08
0.99,5397.99,1.89
0.99,5397.99,1.89
0.98,5389.67,1.70
0.99,5397.99,1.89
0.99,5397.99,1.89
0.99,5397.99,1.89
0.99,5397.99,1.89
0.99,5406.33,2.08
0.99,5406.33,2.08
0.99,5414.66,2.28
0.99,5406.33,2.08
0.99,5406.33,2.08
0.99,5414.66,2.28
0.99,5414.66,2.28
0.99,5423.01,2.47
0.99,5423.01,2.47
0.99,5414.66,2.28
0.99,5406.33,2.08
0.99,5406.33,2.08
0.99,5397.99,1.89
0.99,5397.99,1.89
0.99,5397.99,1.89
0.98,5389.67,1.70
0.98,5389.67,1.70
0.98,5389.67,1.70
0.98,5389.67,1.70
0.98,5389.67,1.70
0.98,5389.67,1.70
0.98,5389.67,1.70
```

REFERENCES

[1] S. Koppelman *et al.*, "Detection of peanut allergens in serum: circumventing the inhibitory effect of immunoglobulins." *Allergy, 75: 1835-1836.*, vol. 17, pp. 132–148, 2020.

[2] M. Xu *et al.*, "Quality tracing of peanuts using an array of metal-oxide based gas sensors combined with chemometrics methods," vol. 128 of *Postharvest Biology and Technology*, pp. 98–104, 2017.

[3] S. Jayasena *et al.*, "Comparison of recovery and immunochemical detection of peanut proteins from differentially roasted peanut flour using elisa," vol. 292 of *PFood Chemistry*, pp. 32–38, 2019.

[4] A. Wesley Burks, "Peanut allergy," *Lancet 2008; 371: 1538–46, Electronically available: https://pubmed.ncbi.nlm.nih.gov/18456104/ [online accessed September 27,2020].*

[5] FARE, "Peanut allergy," https://www.foodallergy.org/living-food-allergies/food-allergy-essentials/common-allergens/peanut, [Online; accessed September 16, 2020].

[6] M. Osman Shabir, "Food allergies explained," https://www.news-medical.net/health/Food-Allergies-Explained.aspx, [Online; accessed December 20, 2020].

[7] W. Loh and M. Tang, "The epidemiology of food allergy in the global context." *Int J Environ Res Public Health. 2018;15(9):2043. doi:10.3390/ijerph15092043.*

[8] M. Calvani *et al.*, "Oral food challenge." *Medicina (Kaunas). 2019;55(10):651. Published 2019 Sep 27. doi:10.3390/medicina55100651.*

[9] I. Chetschik *et al.*, "Quantitation of key peanut aroma compounds in raw peanutsand pan-roasted peanut meal. aroma reconstitution andcomparison with commercial peanut products," vol. 58 of *J. Agric. Food Chem.*, year=2010, pages=11018–11026. DOI:10.1021/jf1026636.

[10] Technopedia, "Data classification," https://www.techopedia.com/definition/13779/data-classification, [Online; accessed December 20, 2020].

[11] R. Garg, "7 types of classification algorithms," https://analyticsindiamag.com/7-types-classification-algorithms/n, [Online; accessed December 20, 2020].