Adaptive Dead Space Enlargement in patients with Central Sleep Apnea Syndrome

Master Internship by Tino Kerkhof, BSc. at Medisch Spectrum Twente.

Contents

Analysis

Problem definition

Sleep apnea

Sleep apnea is characterized by periods of slow or no breathing. Periods with no breathing are called apneas. These periods can last from ten seconds to several minutes and can occur multiple times per hour. When these periods of lowered respiratory function occur, the patient goes from a deep sleep to a more shallow sleep. This reduces the quality of sleep and causes the patient to be tired during the day. This can be dangerous, especially when the person needs to handle heavy machinery or participate in traffic [1]. Sleep Apnea is diagnosed using a Polysomnography. During this test, multiple parameters are measured, like EEG, EMG, arterial oxygen saturation, air flow or volume and snoring sounds. Many more parameters can be added [2].

Sleep apnea can be divided in three classes: central, obstructive and mixed. In patients with central sleep apnea syndrome (CSAS) the signal to breathe ceases. In patients with obstructive sleep apnea syndrome (OSAS) there is an obstruction or extra resistance that prevents air from flowing normally. In patients with mixed CSAS and OSAS, both causes are present. CSAS is associated with heart failure, cerebrovascular accidents and sometimes observed in patients treated for OSAS with continuous positive airway pressure [3].

Breathing

Breathing is controlled by numerous factors. Mechanical feedback is provided by stress receptors in the lungs and thorax, triggered by irritations and mechanical stresses on the pulmonary system. Chemical feedback is provided by peripheral and central chemo receptors, respectively located in the carotid bodies and the medulla. These receptors react on changes in the arterial partial pressure of O_2 (Pa_{O2}), CO_2 (Pa_{CO2}) and pH. The chemical feedback can be characterized by the hypoxic ventilatory response (HVR) and the hypercapnic ventilatory response (HCVR). Both HVR and HCVR are reduced during sleep. Of which the HCVR is especially reduced [4, 5].

Loop Gain

The whole respiratory system can be seen as a complex negative-feedback loop. Controllers in this loop are represented by the mechanical sensors and chemo receptors. The lungs and blood vessels represent the plant. Delays are introduced by the gas exchange of O_2 and CO_2 binding to Hemoglobin, the circulation of air by breathing and the neural transport of information. The main drive of ventilation is the $PaCO_2$. The HCVR tries to keep a stable $PaCO_2$ around the eupneic level (normal Pa_{CO2}) by lowering the air intake when the $PaCO_2$ is too low and increasing the air intake when $PaCO_2$ is too high. The loop gain indicates how strong the system reacts to a disturbance in the $PaCO_2$. A high loop gain in combination with the time delays can lead to instability by overcompensation. This causes hyperventilation as response to high $PaCO_2$ and hypoventilation with or without apneas as response to low $PaCO_2$. The apneic threshold is the Pa_{CO2} where apneas start. A low loop gain is less prone to instability, but can cause prolonged hypoventilation. A smaller difference between the apneic threshold and the normal level of $PaCO_2$ during sleep can also compromise the stability when a slightly high loop gain is present [5].

Stakeholders

Stakeholder	Characteristics	Expectations	Potentials and deficiencies	Implications and conclusions for this project
Patient	Sleeping problem. Not rested after sleep. Social and emotional problems.	A good night sleep, every night.	Potential candidates for clinical trials. Not all patients will respond to treatment.	Potential candidates for clinical trials.
Partner of patient (social circle)	Wants the best and safest solution for the patient. Worries about patient.	Safe solution for the patient.	Can be supportive. Can be protective.	Social support of the patient.
University	Strives to acquire knowledge about the problem and it's solutions.	More knowledge about the problem. New innovative solutions.	Provides knowledge about the problem.	Provide knowledge.
Hospital	Strives to help the most patients as possible.	Solution that reduces the time spend in the hospital.	Provides infrastructure. Bound by strict rules and regulations.	Provides infrastructure for development and testing.
Sleep/lung specialist	Strives for the best solution. Focused on quick and simple solutions.	Easy solution with optimal results.	Expert in patient care. Lack of technical knowledge.	Could be conventional and not use the product.
Technical specialist	Strives for the best solution. Focused on best solution, that may take a long time to develop.	Best solution, optimal results.	Expert in technical knowledge. Lack of experience in patient care.	Could work too long on development.
Society	Demands a lower number of patients, high quality of care, but low cost.	Solution applicable for all patients.	Potential users. Critical in case of failure.	Patient groups could participate in trials, to have better acceptance.
Insurance	Negotiates for optimal care at low cost.	Lower cost for treatment.	Can provide financial support. Only interested if cost effective.	Critical for introduction and a long life time on the market
Industry	Interested in innovative products with low production cost and high sale value.	Profit.	Makes the product and distributes it. Only interested if it makes a profit.	Manufacturer and knowledge of market potential.

Problem

Patient: In patients with CSAS, the loop gain for HCVR is too high and the apneic threshold is too close to the eupneic level of PaCO2. This causes instability in their breathing during sleep. Any disturbance can cause an apneic period. They don't come into deep sleep and are not rested for the next day. This is dangerous when using heavy machinery or participating in traffic. Lack of sleep also impairs the patient's social life and affects his/her emotional state.

Partner of the patient (social circle): The partner of the patient can only watch helplessly as the patient is constantly tired. He/She creates disturbances in bed that can initiate an apneic periods in the patient. The social circle worries about the patient.

University: The university researches CSAS and OSAS and looks for a solution against CSAS and OSAS.

Hospital: Patients with CSAS come into the hospital and need to be diagnosed using a Polysomnography. This means that they need to stay in the hospital overnight. This is very costly. As long as the problem of the patient is not resolved, the patient comes to the hospital more frequently than when it is resolved.

Sleep/lung specialist: The sleep/lung specialist sees the patient at the hospital. He/She has to help the patient get better or cope with his/her problem.

Technical specialist: The technical specialist designs a solution for the problem of the patient. As long as there is no favorable solution, the technical specialist is not satisfied.

Society: There are many patients that suffer from CSAS. All these patients need clinical care. This is mostly paid by society. The patients cannot work as much as they could if they slept normally. This reduces the income of society.

Insurance: The insurance has to pay the costs that the patient makes. If the patient keeps suffering from CSAS, the insurance has to pay more money.

Industry: The industry does not have a clear solution ready to sell. So as long as there exists no solution, the industry does not make money.

Main Cause: The main cause of all these problems is the difference in apneic threshold and eupneic level of PaCO2 in combination with the high loop gain of the HCVR and/or a high time delay of the control loop. This causes the instable breathing pattern of the patient. Any disturbance in breathing can cause apnea in the patient.

Goal

Make the breathing pattern of patients with CSAS more stable to let them fall into deep sleep.

This can be done by fulfilling any or a combination of the following goals:

- Increase the difference between the eupneic level of PaCO2 and the apneic threshold.
- Lower the loop gain of the HCVR.
- Lower the time delay of the loop.

Assignment

Design a machine that can actively and adaptively enlarge the dead space to increase the carbon dioxide levels in the longs and blood to let the body of patients with Central Sleep Apnea Syndrome (CSAS) keep giving signals to breath.

The material cost of all concepts and the final concept before testing the proof of concept may not exceed €300,-.

List of requirements

Use requirements

- The product must be able to measure the partial pressure of CO2 in the expired air with a range of at least 0 – 15 kPa with a resolution of at least 0.1 kPa.
- The product must be able to measure air intake, output and flow of breathing.
- The product must be able to change the amount of CO2 in the inhaled air by 0 – 25 ml per breath with a resolution of at least 5.0 ml.

Safety requirements:

- The product may not prevent the patient from breathing.
- The product may not lead to hemodynamic instabilaties.
- The product must withstand moist from breathing.
- In case of power failure, the patient must be able to keep breathing with a tidal volume of 750 ml/breath or less.
- The product may not conduct electricity to the user.
- The product may not be poisonous to the patient.
- The product may not cause the CO2 in the inhaled air to exceed 10 kPa.

Ergonomics requirements

- The product interfacing with the user should weigh less than 500 g.
- The product should not cover the eyes.
- The product should not apply more pressure on the skin of the head than a standard Mouth/Nose mask used for CPAP.
- The product should not dry out the airways.
- The product should not go into the user further than 2.0 cm into the mouth or 0.5 cm into the nose

Space requirements

- The product interfacing with the user should fit on the nose and mouth region of the user.
- The product should not be higher than 100 cm.
- The product should not be wider than 40 cm.

Time requirements

- The product must last for 10 years.
- Maintenance is required no more than four times per year.
- The product must be calibrated to the patient within one hour of sleep.
- When calibrated, The product should fulfill its goal within 5 minutes after disturbance.
- Daily cleaning should take no longer than 15 minutes of the user’s time.
- Monthly cleaning should take no longer than 1 hour of the user’s time.

Use wishes

- It must be possible to control the product after an explanation of the doctor of at most 15 minutes.
- Maintenance is required only two times per year.
- Daily cleaning should take no longer than 5 minutes of the user's time.
- Monthly cleaning should take no longer than 15 minutes of the user's time.

A commonly used treatment for OSAS is continuous positive airway pressure (CPAP) and since most patients with CSAS also have a form of OSAS, it is favorable to incorporate CPAP into the design. CPAP enlarges the physiological dead space. This has to be taken into account when designing the product in combination with CPAP [5].

When the P_{CO2} in the inhaled air is increased, respiratory acidosis in the blood and brain is immediate. When the Pa_{CO2} stays elevated, the pH in the cerebrospinal fluid (CSF) and brain extracellular fluid (BECF) slowly increases over 8-24 hours because of an increased influx of $HCO3^-$ into the CSF and BECF. This increase in pH with a higher constant PaCO2, shifts the HCVR to a higher PA_{CO2} and decreases it's gain. This increases stability of the respiratory system, but the HCVR may be shut down completely and breathing will only be controlled by the HVR if the HCVR shifts too much [6].

Function Analysis

Main function

The main function of the product is adapting the CO_2 in the air. Through the adaptation of the CO_2, the Pa_{CO2} changes and the breathing pattern of the user is changed.

Initiation and closing

When the user is going to use the product, the product needs to be interfaced with the user. Next the user switches the product on and fills in the parameters for the next use. The product is switched off and the interface is removed again when the user is done.

Acquire information

The P_{CO2} in the breathed air is measured and the flow of the breathed air is measured.

Process information

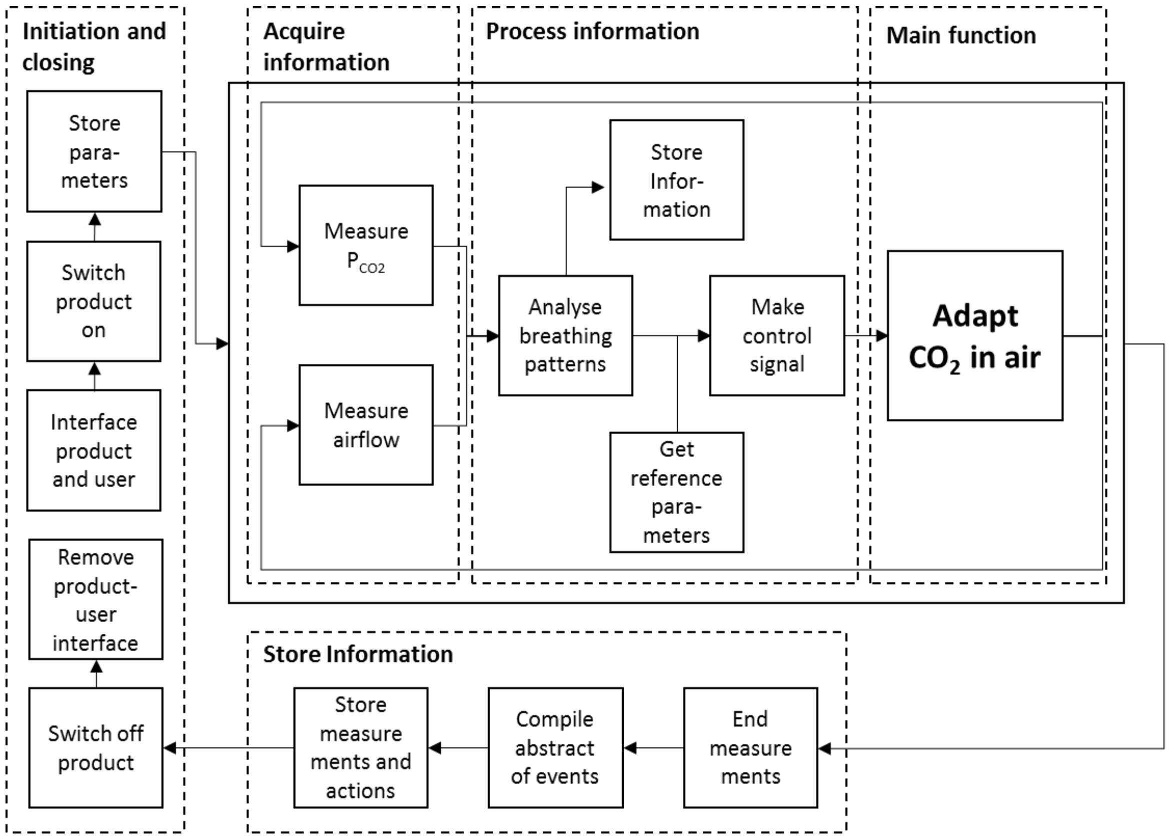
The P_{CO2} in the air and the airflow are combined to analyze the breathing pattern. This information is stored. With this information, the necessary adjustments the product has to make are estimated and a control signal is send.

Store Information

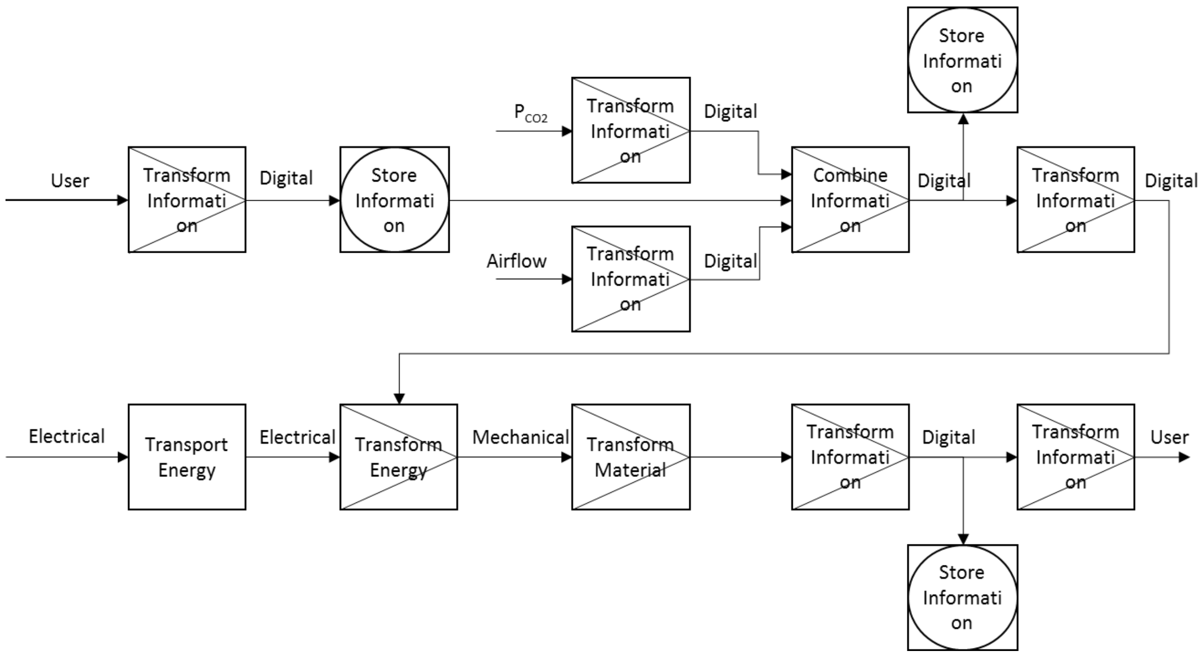
At the end of a session the measurements are ended. An abstract of all the relevant events are compiled and the measurements and actions are stored.

Function scheme

Flow chart



Function scheme



Synthesis I: Initial concepts design and selection.

Readily available solutions

Digital information

Because the product will be designed for a preliminary study it is important that all the digital information can be stored and reviewed later. Therefore all processes are run through a computer using the program Matlab®, program developed by MathWorks, Inc..

Measurement of P_{CO2} and breathing.

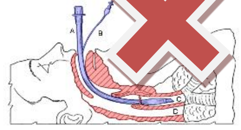
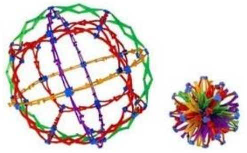
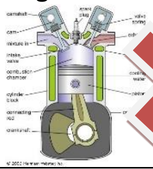
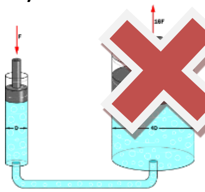
Volumetric Capnography (VCap) measures the concentration CO2 in the expired air. The dead space, Pa_{CO2}, end-tidal CO2 (PET_{CO2}) and more parameters can be calculated with the VCap. These parameters can be used to measure the regulation of breathing [7] .

A capnograph is already available for use. The CO2SMO plus!® capnograph, developed by the firm Novametrix Medical Systems Inc.. This system can measure P_{CO2}, air flow, pressure and pulse oximetry. With these parameters it calculates the inhaled volume. These parameters can be used to analyze the breathing patterns. The main drawback of this system: The data acquired by the software of CO2SMO plus!® can only be analyzed offline after the measurement. It may be possible to bypass the software using Matlab®.

If it is possible to bypass the software of the CO2SMO plus!® using Matlab®, all digital functions have initial solutions. So only initial designs of the adaptation of the P_{CO2} and the physical user-product interface are left.

Morphological scheme

Table 1: Morphological scheme

Function \ Solution	1	2	3	4	5	6	7	8
User-Product connection	Nose mask	Nose mouth mask	Full head encasement	Nose tube	Endotracheal tube 			
Air supply	Air tank	The air	Tubes	Fans	CPAP			
Adapt CO_2 in air	CO2 from compressed air tanks	Adapt dead space Compressible tubes	Adapt dead space Trombone sliding system	Adapt dead space 3 way valves	Adapt dead space Y pieces	Adapt dead space Hobberman sphere 	Adapt dead space Full head encasement	Adapt dead space Trumpet
Transform Energy	Electromotor	Combustion engine 	Solenoid					
Transform Material	Rails	Gears	Joints	Hydraulics 				

The Morphological scheme in Table 1 shows some ideas about how to adapt the CO2 in the air. These ideas were gathered by a brainstorm. The main function is 'Adapt CO_2 in the air', the selection of sub-solutions for the pre-concepts starts in that row.

Air supply

Air supply solution 'The air' is used in all other solutions as well, as dilution or as the main supply for air. So 'The air' is removed from the morphological scheme.

'Fans' by themselves are not effective in supplying air. Inside a tube, they create suction at one end of the tube and positive pressure on the other end. This way they can deliver air efficiently. This positive pressure is used by CPAP. So in the Morphological scheme, 'CPAP' is added instead of 'Fans'.

CO_2 from compressed air tanks is not preferable to use, because the safety requirements are hard to accomplish. If there is a failure the maximum CO_2 in the air is easily exceeded. The risk is too high. Therefore 'CO_2 from compressed air tanks' is removed from the morphological scheme (Also in the row 'Adapt CO2 in the Air').

Adapt CO2 in the air

The full head encasement is not a save method to adapt the CO2 in the air. Although the CO2 in the air is quickly changed by turning off the air supply to the head and create rebreathing, the air supply is active. In case of power failure the user would quickly suffocate. Full head encasement is also removed in the 'User-Product connection'.

User product connection

All remaining solutions for adaptation of CO_2 in the air are based on the production of CO_2 by the body of the user and use an increased dead space to adapt the CO_2 in the inhaled air. It is crucial to have no leakage of air when using this method, because leakage let's CO_2 out and O_2 into the dead space. The 'Nose tube' does create extra dead space. When a user with a 'Nose mask' breaths through the mouth, the enlarged dead space at the nose is not used. The 'Nose tube' and 'Nose mask' are removed from the morphological scheme.

The 'Endotracheal tube' does not fulfill the ergonomic requirements. It is too invasive: It enters too far into the mouth of the user. It is removed from the morphological scheme.

The only idea left is the 'nose and mouth mask'. This will be the User-Product connection.

Transform Energy

The 'combustion engine' will not be used here, burning fuel inside a house is not healthy. There are cleaner energy sources available: electricity.

Transform Material

Hydraulics will not be used for this product. Although it is a good and solid way to transform material, the design would very quickly become too complex and expensive.

Pre concepts

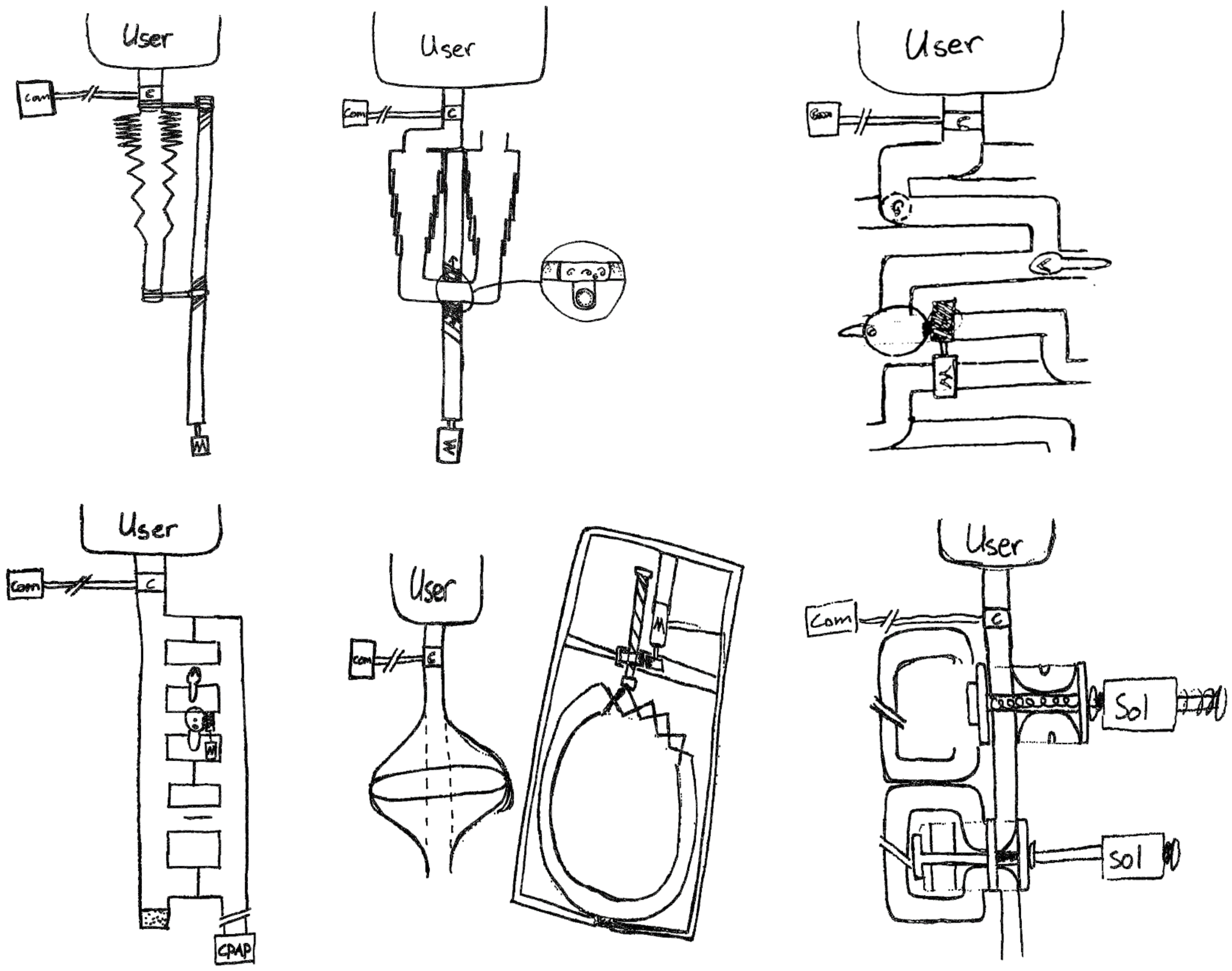


Figure 1: Concepts from Morphological scheme: In reading order: Compressible tubes, Trombone, 3-Way valves, Y-pieces, Hobberman sphere, Trumpet.

Compressible tubes

For this concept, compressible tubes, gears and an electro motor are used. Extra dead space is added by pulling the end point and unfolding the compressible tubes. The location of the endpoint of the tube can be very accurately measured, but the exact volume of the tube not, because the way the tube compresses and unfolds cannot be controlled or predicted. CPAP can be applied.

Trombone sliding system

This concept consists of trombone sliding system, rails and an electro motor. Extra dead space is created by sliding the tubes out. Location of the midway point can be accurately measured, the volume can be just as accurately estimated. The midway point does not give the volume, the way it got there has to be taken into account when calculating the volume. The drawback for this concept is that it becomes bigger when more tubes are added, because every tube has to fit inside the next. CPAP can be applied.

3-Way Valves

This concept consists of 3-way valves, tubes and joints. The extra dead space is added by closing more valves and making the path to the outside air larger. The extra dead space can be accurately estimated, but is increased in increments. CPAP cannot be applied.

Y-Pieces

This concept consists of Y-pieces, electro motor, tubes, joints and CPAP. The extra dead space is added by opening a one way valve further away from the mouth entrance. The extra dead space can be accurately estimated, but is increased in increments. CPAP is required.

Hobberman sphere

This concept consists of the Hobberman sphere, electro motor and gears. The extra dead space is added by mechanically increasing the size of the sphere. The dead space can be accurately estimated by the measure of the diameter of the sphere. The sphere is very complex though. CPAP can be applied.

Trumpet

This concept consists of Trumpet, solenoid, joints and tubes. The extra dead space is created by increasing the path the air has to take from the mouth to the end of the tube by pushing in the diverted way. The dead space is increased in increments, but there can be more increments than in the '3-way valve' concept, because the extra dead space can be increased using a binary system. The first diversion adds 1, next diversion adds 2, next adds 4, and so on... By making combinations small increments can be achieved. CPAP can be applied.

Pre concept selection:

To select the concept for further detailing, a comparison is done. It is yet unknown how each concept would score on the different product requirements so the factors to weigh in are simplified.

Minimal extra dead space: The extra dead space that the concept adds by just being present should be as low as possible. If this is too high, the CO2 in the lungs goes up even without intervention.

Resolution: The extra dead space should be added in the lowest possible increments. If the increments are too high, the control system becomes less accurate.

Complexity: The product can have some complexity to fulfill its use, but if it becomes too complex, it is less likely to be finished in the timespan allotted to this assignment.

Concept	Minimal extra dead space	Resolution	Complexity
Compressible tubes	-	++	++
Trombone sliding system	-	++	+
3-way valves	++	-	+
Y-pieces	++	-	+/-
Hobbermann sphere	+/-	++	--
Trumpet	+	+	-

In this comparison, the Hobberman sphere and the Trumpet, perform the worst. They are too complex to build in a short amount of time. And the benefits to the performance, because of this complexity, are not enough to compensate.

The 3-way valves and the Y-pieces perform best in terms of minimal extra dead space, because the closest valve to the mouth can return the added extra dead space to a minimum. The concept Y-pieces scores lower on complexity, because CPAP is required.

The Compressible tubes and Trombone sliding system are the least complex and the dead space can be adjusted with high resolution. But they both add extra dead space by just being used.

The concepts that are selected for detailing are the Compressible tubes and the 3-Way valves.

Synthesis II: Concept detailing.

Normally at this point in the design process, the selected concepts are detailed and then scored to the requirements to make a well informed decision about the further process of development. But because of time constraints within this assignment it is desided to combine the concepts chosen in the last chapter and work out one concept.

The positive part of the compressible tubes concept is its resolution and the positive part about the 3-way valve is the small extra dead space added by just being used. To combine these two concepts, a 3-way valve will be put between the capnograph sensor and the compressible tube. In this new concept the standard added dead space will be minimal and when adding dead space, the resolution will be high.

Adaptive Dead Space Enlargement

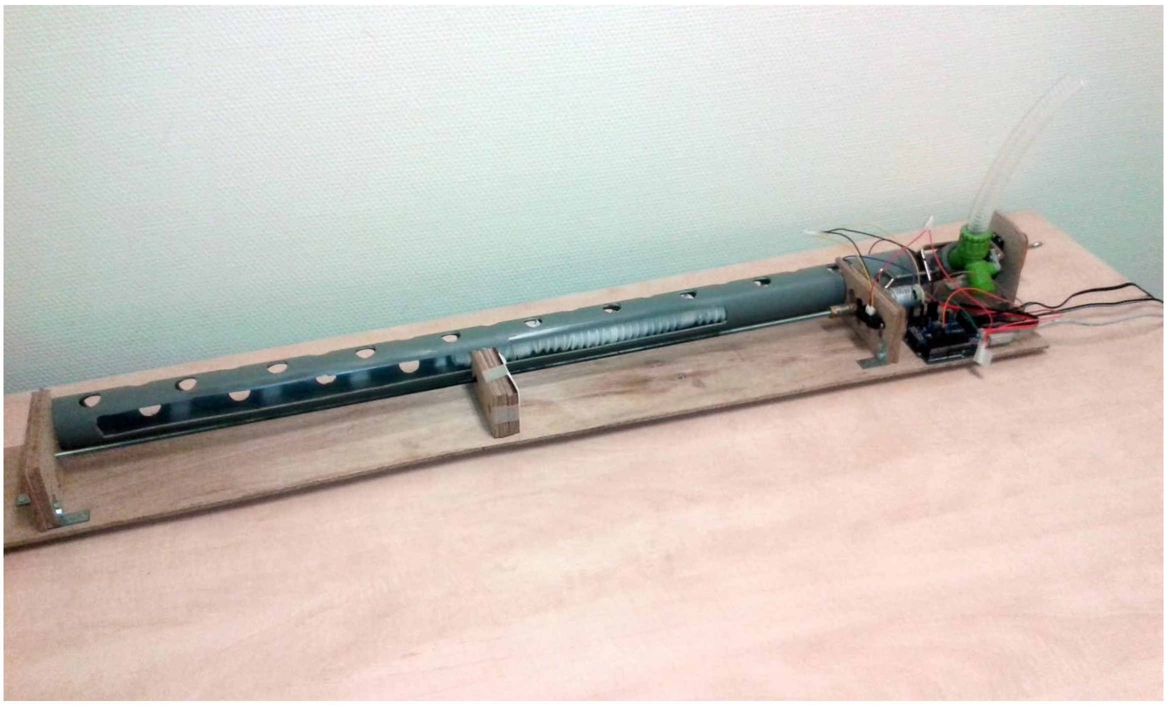


Figure 2: Adaptive Dead Space Enlargement

Mouth/Nose mask

The Mouth/Nose mask is a standard Mouth/Nose mask. No changes have been made to the mask. The white plastic object right after the mask is an air filter. The product will first be used for research purposes and will therefore be used by multiple people. This filter cleans the air as much as possible. The mouthpiece that comes with the filter is cut to fit on the Mouth/Nose Mask.

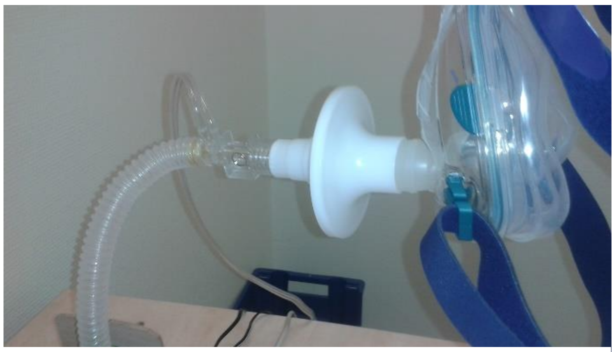


Figure 3: Mouth/Nose Mask

Capnograph

The sensors of the Capnograph come next. These sensors measure air flow and CO_2 concentration in the air. More on this later.

Flexible bridge

The flexible bridge allows the user to move his/her head without moving the whole product.

3-way valve

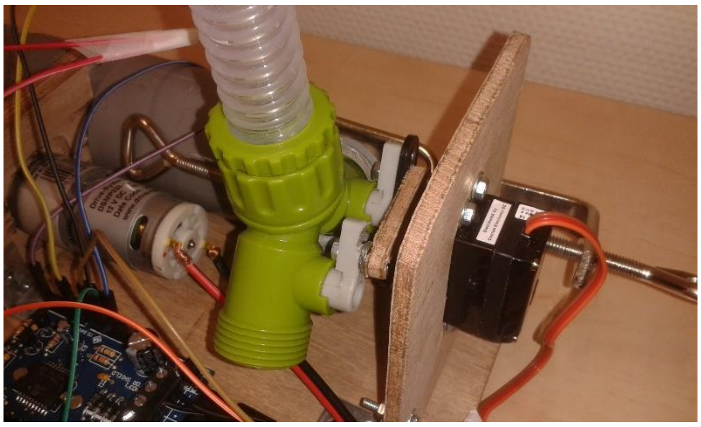


Figure 4: 3-Way Valve

This is the part of the 3-way valve concept. This 3-way valve is build up from a Y-piece with two 1-way valves. When both valve handles are turned the same way, one valve is open and the other is closed. This makes it possible to connect the handles with a rigid bar, like wheels of a train, to turn both handles in the same direction when turning one handle. A servo-motor is used to turn the handles automatically. A servo-motor is a DC-motor with an encoder that records the angle of the last gear and controls this angle according to a received digital signal.

The dead space from the mouth until the end of the 3-way valve, when it is turned to the outside air, is about 100 ml.

Compressible tube

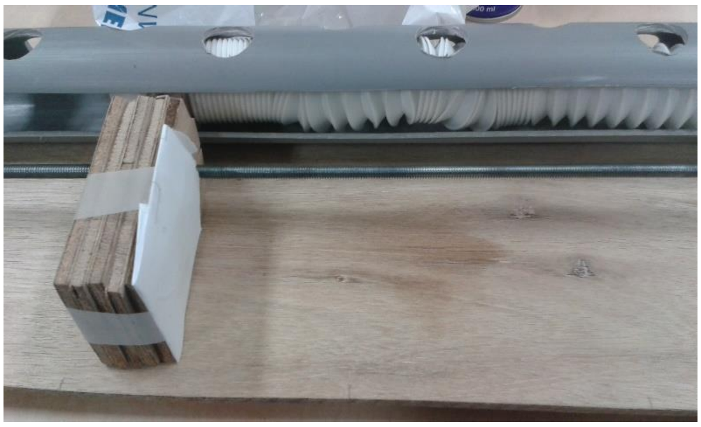


Figure 5: Compressible tube and guide tube

This is the part of the Compressible tube concept. When the 3-way valve is turned to add extra dead space the user breaths in and out through the compressible tube. This tube can be compressed to be about 170 ml in volume and be extended to be about 550 ml in volume. The tube does not behave ideally though. When the tube is extended the whole tube is stretched. When the force of the pull exceeds a certain limit, one of the folded sections unfolds and the rest retracts a bit and this process gets repeated. The force of the pulling needs to be quite high. The same happens the other way around, but the refolding does not happen over the whole circle at once, but in 2-3 steps. This causes the compressible tube to bend.

Guide tube

To prevent this bending a guide tube is made. This is a rigid tube that prevents the compressible tube to bend too far away. This tube may not add extra dead space, so numerous holes are made along the tube to let the breathed out air out and new air in.

Energy transfer

The compressible tube is extended and retracted using a DC-motor. This motor turns a threaded wire. This threaded wire is fixed in position parallel along the guide and compressible tubes. By turning the wire a block with two nuts in it is moved along it. This block in turn then pulls and pushes the compressible tube.

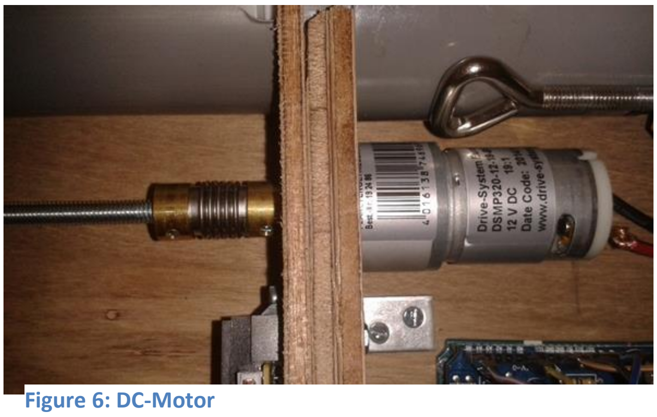


Figure 6: DC-Motor

Proximity sensor

The proximity sensor measures the distance from the sensor to the first object in its line of sight, in this case the block that pushes and pulls the compressible tube. The signal of the proximity sensor is not linear, so a translation is made. The distance from the sensor to the block in increments of 2 – 2.5 cm from minimal added dead space until maximal added dead space is matched with the signal the sensor returns. A polynomial expression for the total Added Dead Space as a function of the sensor data is made.

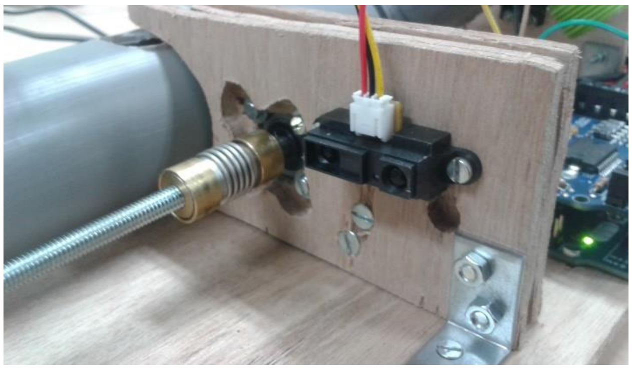


Figure 7: Proximity Sensor

Arduino UNO

The Arduino Uno is an open source computer board with a microcontroller that can process input and output from a number of digital and analog pins. These pins can be used to provide electricity to small machines, control small to medium machines and receive and process sensor data. For this product The Arduino Uno is used in several ways. It provides power to the servo motor, proximity sensor and a button. It receives signals from the sensor and a button. It sends signals to control the power delivered to the DC-Motor. And it sends signals to control the direction of the DC-Motor and the target position of the servo-motor. The regulation of electricity for the DC-Motor is done via another computer board, the Velleman Motor Shield. This shield has some additional electrical components, that allows the Arduino to regulate the power provided to the DC-motor and therefore how fast it turns. It also has safeguards that prevent back current.

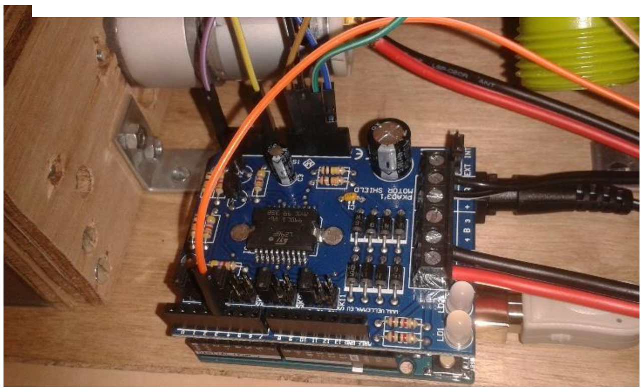


Figure 8: Arduino Uno

The Arduino can be connected to the computer and communicate with Matlab®. The Matlab code to control the Arduino Uno can be found in the appendix.

Structure

The structure of the product will be made out of multiplex. Multiplex can easily be shaped. It can be voluptuous though.

Capnograph CO2SMO plus!

The CO2SMO plus!® is a capnograph. It can measure the CO_2 concentration in the air, the air flow, air pressure and pulse oximetry. With this data it can calculate other data. Like expired volume, volumetric capnogram, respiratory minute volume, etc..

The CO2SMO plus!® capnograph interacts with the computer via a RS232 connection using a computer program to command the capnograph. The computer program commands the capnograph to send the data to the computer, reads and interprets it and shows the results live on screen. Timon Fabius, a colleague intern and master student Technical Medicine, previously analyzed the data saved by this program and established that the data can be split in five signals: Flow, Volume, Expired Volume, P_{CO2} and Pressure. The program is sufficient when analyzing the data offline. But the product we are developing needs the data from the capnograph immediately to control the dead space with a relatively small time delay. Therefore it is needed to read the data from the capnograph outside of the computer program of Novametrix.

Matlab® is used to interact with the capnograph, interpret and analyze the data. To achieve this, a number of questions must be answered:

- What are the specifications of the connection between the program and the CO2SMO plus!®?
- What language is used by the program and the CO2SMO plus!® to interact?
- What data type is used by the CO2SMO plus!® to express the data?
- How is the data ordered?
- In what units are the different signals expressed?

Specifications

On the website of Phillips, Phillips is the current developer of the CO2SMO plus!®, the only specification of the connection between the computer and the CO2SMO plus!® is the use of a RS232 connection. The standard serial parameters used by Matlab® do not give a desired response. When asking for the identity of the CO2SMO plus!® using the standard command '*IDN?', it sends '* * * * *i' back. Any other command sends a series of '* ' finished with '*i' back.

A monitoring program, Device Monitoring Studio (DMS), is used to analyze the connection and the data send between the computer and the device. DMS registered the changes in the connection settings the program of the CO2SMO plus!® made. It changed the Baud rate from 9600 to 19200. All other parameters stayed unchanged.

Language

DMS showed the data send by the program of the CO2SMO plus!® to initiate the connection, maintain idle connection, start a measurement and end the measurement. In ASCII these signals are gibberish. The signals, send by the program of the CO2SMO plus!®, were copied to a text document. In Matlab® the text was translated into a series of ones and zeros, expressed in unsigned 8 bit integers, that could be send to the capnograph. When the signals were send from Matlab®, the answers from CO2SMO plus!® were the same as when the signals were send using the program of the CO2SMO plus!®.

Data type

Timon Fabius established that the data, saved by the program of the CO2SMO plus!® is expressed in 16 bit signed integers, so Matlab® was instructed to interpret the data as 16 bit signed integers. The data from the CO2SMO plus! was different from the saved data, so it can be concluded that the program of the CO2SMO plus! does some post processing before saving the data. The data from Matlab® shows most values between zero and 2^{15}. There is just one signal below zero. The data directly from the CO2SMO plus! could be either signed or unsigned.

A measurement was done with Matlab®. During this measurement DMS recorded the incoming data. From the data of DMS clear signals can be distinguished, while the data from Matlab® are not so clear. After changing the endian of the Matlab® data, the data was the same as the data from DMS. Figure 9 and Figure 10 show this change.

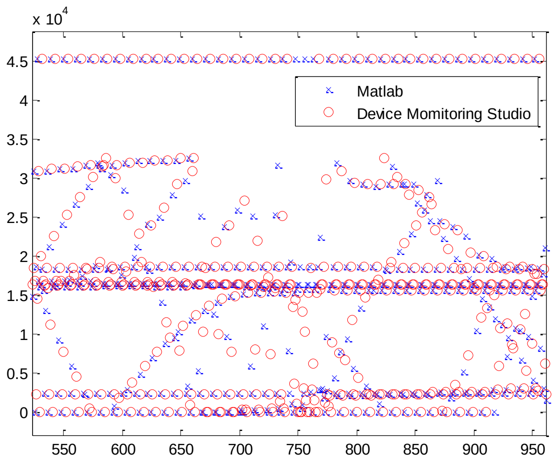


Figure 9: Data before Little/Big endian swap.

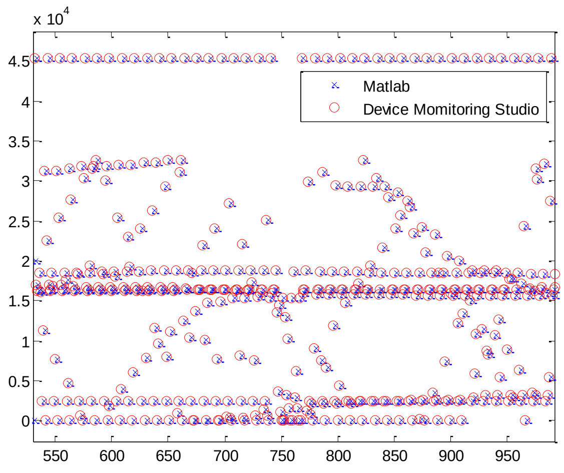


Figure 10: Data after Little/Big endian swap.

Ordering of the signals

The data is read as a vector of values. The values of the distinguishable signals are spaced eleven data points from each other in most of the data. Because this is not the case for the whole signal, a signal that is clearly distinguishable from the others is used to sort the rest of the signals. This signal is the only signal with negative (or values above 2^{15}). Eleven signals are made, taking the ten data points around this point and coupling them. Figure 11: The 11 signals.Figure 11 shows the 11 signals.

Signals

Signal 7 is the signal below zero or above 2^{15}. It is not known what this signal represents. Signal 8 9, 10 and 11 are most clear. They all have an offset of 2^{14}, this can be interpreted as zero, signal below 2^{14} is negative, signals above 2^{14} are positive. This offset is removed by subtracting 2^{14} from the signals. Gaps in the signal values can be seen. The gaps are 128 high and the spaces with data are also 128 high. The 8th bit seems to be unused or reserved. These gaps are removed by subtracting a correction from each point. This correction is dependent on the value of the data point.

$$DataCorrected = Data - \left(\frac{floor\left(\frac{Data}{128}\right)}{2} * 128 \right)$$

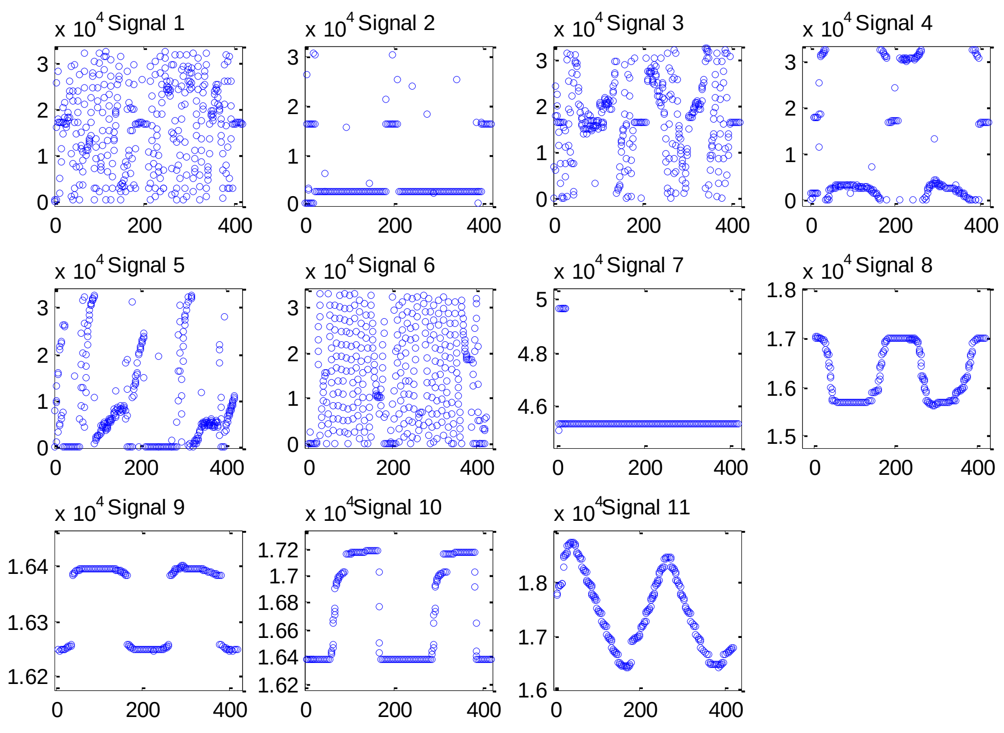


Figure 11: The 11 signals.

Units

After these corrections, the units of the values are established. We know that the data from Matlab® and the data from DMS are the same, so DMS can be used to compare the signals from Matlab® with the signals from CO2SMO plus!®.

Signal 8 corresponds with the flow. The values from CO2SMO plus!® for flow are 10 times smaller than signal 8 from DMS, so the resolution for flow is 0.1 L/min.

Signal 9 corresponds with the pressure. The values from CO2SMO plus! for pressure in the air are 75 times smaller than signal 9 from DMS, so the resolution for pressure is 0.013 kPa or 0.1 mmHg.

Signal 10 corresponds with P_{CO2} in the air. The values from CO2SMO plus! for P_{CO2} in the air are 75 times smaller than signal 10 from DMS, so the resolution for P_{CO2} is 0.013 kPa or 0.1 mmHg.

Signal 11 corresponds with the inhaled volume. The values from CO2SMO plus!® for Volume are the same as the values from DMS. The resolution for Volume is 1 ml.

The Matlab® code that reads and processes the CO2SMO plus!® data can be found in the appendix.

Synthesis III: Final concept realization.

Prototype construction

Appliances

The prototype for the adaptive dead space enlargement is made using basic tools:

Automatic screw driver/drill	Hand saw
Drills	Hand metal saw
Screw driver	Fret saw
Bahco	Tape measure
Hamer	PVC glue

Materials

Materials used to make the prototype:

Dead space	Framework	Electrical components
Nose/Face mask	Multiplex	Arduino UNO Development Board
Air Filter	PVC tube	Arduino Motor Shield Velleman VMA03
Tubes	Bolds and nuts	Driver System Europe DC-Motor DSMP320-12-19-B-F
2 way valve	Rubber bands	Bluebird BMS-410
Compressible tube	Flexible connection	Sharp Proximity sensor GP2Y0A21YK0F
	PVC connections	USB cable
	M6 Tread wire	9V Adapter
	M6 Bolds	12V adapter
		Electrical wires
		Isolation tape

Final concept detailing

The Adaptive Dead Space Enlargement device and the CO2SMO plus!® must now be combined to adapt the dead space based on the measured capnograph data.

Parallel computing

The CO2SMO plus!® and the Arduino are both connected to the computer and read and controlled using Matlab®. It is sufficient to read the data of the CO2SMO plus!® every 4-6 seconds to register every breath. The analysis of the data from the CO2SMO plus!® is done within that time. The Arduino controls the Dead Space Enlargement with a DC-motor and a proximity sensor. The position of the block that moves the endpoint of the compressible tube needs to be watched closely for the best control. This is especially the case for minimal and maximal added dead space. The time between two moments of sensing the position of the block is preferably lower than half a second. Reading the data from the CO2SMO plus!® and analyzing the data takes longer than half a second.

For this reason it was decided to use parallel computing to control the dead space and read the CO2SMO plus!®.

The system was split in two parts. Sensing CO_2 together with analysis of the data and control of the added dead space.

Communication

When using parallel computing, Matlab® sends the different tasks to different workers, processors. These workers perform the given tasks separated from each other. There is no immediate communication between the two, variables made in one worker are not made in the other. There are however ways to actively communicate data from one worker to the other using a send and a receive function.

The send function sends data to a specified worker and immediately goes on with the next command. It does not wait for the other worker to receive the data. The receive function receives data from a specified worker, but waits for that data to be send if it was not already send before moving on to the next command. To prevent delay, a third function is used to probe if there is data available to receive. Only if there is, the receive command is given.

User interface

It was also tried to make a user interface for the purpose of stopping the system in a controlled manner where all the data would be saved and not cleared, which is the case when using a hard interrupt (Ctr-C). This is sadly not possible, because the worker cannot interact with the client, Matlab®'s main process space, yet. It can send data to print in the command window, but it cannot receive data that is inputted.

There may be one more solution to make a user interface. Use another client of Matlab® to save the user input in files and then read the files from the workers. This is not user friendly, because you have to start up and use two programs at the same time, and reading from the computer's hard disk takes time.

Instead of the user interface a button is made and connected to the Arduino to signal 'Stop'.

$PETCO_2$

The Inhaled Volume and CO_2 data from the CO2SMO plus!® are used to calculate the $PETCO_2$. The Volume data is used to extract the begin and end points of each breaths. The highest CO_2 level between the top of the inhaled Volume and bottom of the Inhaled Volume is used as $PETCO_2$. Breaths smaller than 150 ml are not taken into account, because the PCO_2 inside the anatomical dead space is not representable for the $PaCO_2$. Also $PETCO_2$ below 1 kPa is not taken into account. This is to filter the measurement stops of the CO2SMO plus!®. These measurement stops occur when the CO2SMO plus!® cleans the pressure sensors. During this stop, the signals for pressure, flow and volume are put to zero. If this happens during the first 100 ml of exhalation, the measured CO_2 is zero.

Dead Space

The dead space required for the control of the $PETCO_2$ is found using Proportional Integral control. The proportional control adapts the added dead space proportionally to the error between the measured $PETCO_2$ and the reference $PETCO_2$. The integral control adds or removes dead space from the added dead space according to the error between the measured $PETCO_2$ and the reference $PETCO_2$. The error proportional for the proportional control is raised to the power of 3 and the error for the integral control is raised to the power of 1/3. This is done to make the dead space adapt to small errors using integral control, and to big errors using proportional control.

If the reference $PETCO_2$ is reached, the added dead space does not change. If the reference $PETCO_2$ is uncomfortable for the user, he/she will instinctively breath more, lowering the $PETCO_2$, causing the added dead space to increase. This can cause the user to hyperventilate, while maintaining the correct $PETCO_2$. To prevent this, for every breath a certain amount of dead space is retracted from the integral part of the control. A positive side effect of this: The system tries to find the lowest added dead space to keep a stable $PETCO_2$. A negative side effect of this: The $PETCO_2$ will be below the reference $PETCO_2$ more than above it.

The formulas below explain this in mathematical terms.

$$\Delta PETCO2 = PETCO2Ref - PETCO2$$

$$P_{ADS}(n) = P * \Delta PETCO2^{3}$$

$$I_{ADS}(n) = I_{ADS}(n-1) + I * \Delta PETCO2^{\frac{1}{3}} - AHV$$

$$Added\ dead\ space\ (n) = I_{ADS}(n) + P_{ADS}(n)$$

Results

The workers can only send data, to be printed in the command window, to the client. They cannot do other things on screen. This includes plotting. For this reason, the results are not shown during the use of the product. The results are saved to the computer and shown when both parallel paths are finished.

The results shown are: Air Flow, Inhaled Volume, Pressure, PCO_2, $PETCO_2Ref$, $PETCO_2$, added dead space and target added dead space. All against the time.

The Matlab® code can be found in the appendix.

Prototype testing

With the Adaptive Dead Space device, the CO_2 sensor and the control software are all put together and tested. Energy is provided to the Arduino, the DC-Motor (via the Arduino), The CO2SMO plus!® and the computer. The Arduino and the CO2SMO plus!® are connected to the computer. A test subject put the Mouth/Nose mask on and breathed through the device. Then the software is turned on.

Before starting, the place to save the data is and the target $PETCO_2$ are put in.

First thing after starting, the program makes the save directory, starts the measurements and moves the Added Dead Space to minimal.

When the first measurements are done, the capnograph software calculates the Added Dead Space and sends this to the Arduino Control software. The Arduino software adapts the Added Dead Space and the capnograph continues measuring.

When the timeperiod of the test is over or the stop button is pushed, the Added Dead Space is turned to minimal and the data is saved. After this the data is showed in plots.

The Test:

The test subject starts with normal breathing for about one minute. After that minute the test subject starts to hyperventilate, higher breathing frequency and higher tidal volume. After about two minutes of this, or when the test subject thinks it's enough, he breaths normal again.

Figure 12 and Figure 13 show the results of this test. Figure 12 shows the capnograph data and Figure 13 shows the Adaptive dead space data.

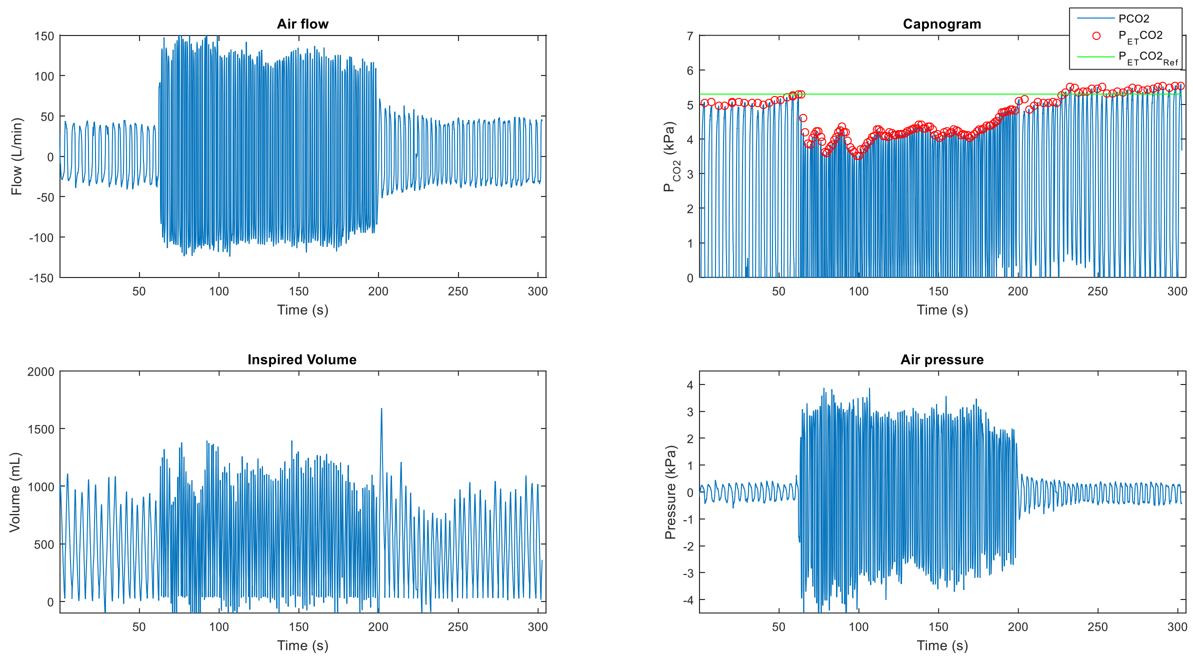


Figure 12: results of test: breathing

Figure 12 shows: in the first part of the test, the $PETCO_2$ is stable approximately 0.1 kPa below the desired $PETCO_2$. In the second part the $PETCO_2$ drops quick, but does not drop further than 3.0 kPa. After 165 seconds, the $PETCO_2$ starts to rise, even though the test subject still hyperventilates. The $PETCO_2$ returns to the target almost immediately after stopping hyperventilation, but overshoots a little. Figure 13 shows that in the first part the target dead space stays around the starting target dead space. The 3-way valve is opened when the target dead space drops below the lowest possible dead space with the 3-way valve closed. In the second part, the target dead space is increased very fast, because the $PETCO_2$ drops fast. This is because of the proportional control. The actual dead space takes about 100 seconds to go from minimal to maximal Added Dead Space. After the hyperventilation stops, the target dead space drops. Not to minimal Added Dead Space, but to about 2/3 between minimal and maximal. This is because the integral part of the control also increased the target over time. The $PETCO_2$ was above the target $PETCO_2$, so the target Dead space dropped slowly until the test was ended.

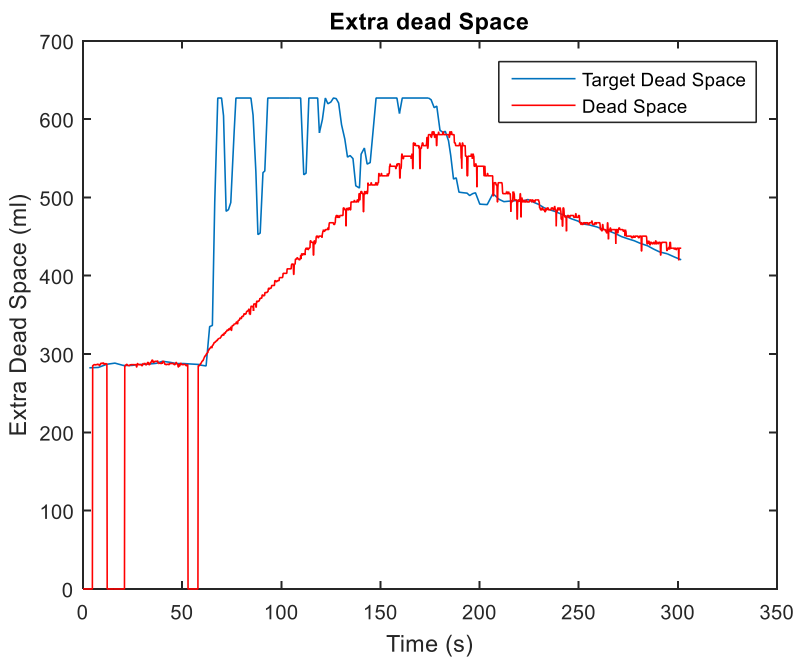


Figure 13: results of test: Added Dead Space

The behavior of the device was as intended.

Recommendations

The first impression of the workings of the product are good, but more testing needs to be done to test the abilities of the product.

There may be more parameters with which the breathing can be controlled. It may be interesting to inspect the inhaled volumetric capnogram. The inhaled VCap shows how much CO_2 is rebreathed. Together with the normal VCap the net exhaled CO_2 can be calculated.

The testing of the product can be done by either a Technical Medicine student or a Biomedical Engineering student. At this stage I do not recommend a student with little or no experience using Matlab.

If the tests have a positive outcome the product can be optimized to better fit the requirements. An Industrial Designing student can design a product with custom made components and an Electrical Engineering student can minimize the electrical components of the design. Their design will be more expensive in production.

Some examples of aspects which can be optimized:

- A less stiff compressible tube. This allows a weaker motor or a quicker energy transfer system.
- A bigger and safer 3-way valve. The servo motor does not open the valve to the open air in case of a power failure. The 3-way valve used in this product has a narrow airway, which causes high breathing resistance. A bigger valve will give less resistance.
- A lighter and smaller frame. This is ergonomically better for the user.

References

1. NHLBI. *What Is Sleep Apnea?* 2012 [cited 2015 05-02-2015]; Available from: http://www.nhlbi.nih.gov/health/health-topics/topics/sleepapnea/.
2. Grinten, C.v., *Poly(somno)grafie: sensoren en meettechnieken.* Slaapcursus der lage landen voor artsen deel 1. **1**: p. 201-217.
3. Morgenthaler, T.I., et al., *Complex sleep apnea syndrome: is it a unique clinical syndrome?* Sleep, 2006. **29**(9): p. 1203-9.
4. Oostveen, E., *Fysiologie van de bovenste luchtwegen bij OSA.* Slaapcursus der lage landen voor artsen deel 1, 2015. **1**: p. 25-34.
5. D.G. Mc Sharry, D.J.E., A. Malhotra, *European Respiratory Monograph 50: Sleep Apnoea.* 2010. p. 381-395.
6. Boron, W.F. and E.L. Boulpaep, *Medical Physiology: A Cellular and Molecular Approach*. 2003, W.B. Saunders. p. 712-734.
7. Suarez-Sipmann, F., S.H. Bohm, and G. Tusman, *Volumetric Capnography: the time has come.* Current Opinion, 2014. **20**(3): p. 333-339.

Appendix

Matlab scripts

Main.m

```
function data = Main
% pre clean-up
close all; clear all; clc;
if isempty(instrfindall) == 0
    fclose(instrfindall);
end
delete(gcp('nocreate'))

% Chose save directory
sdir = 'Data/FILLINSDIR/';
data1.sdir = sdir;
data2.sdir = sdir;
if exist(sdir,'dir') == 0
    mkdir(sdir);
else
    OW = input('Map already exists, do you want to stop? [1/0], [0]: ');
    if OW == 1
        return
    end
end

% Put in the COM port names
data1.CapnoCOM = 'COM2';
data2.ArduCOM  = 'COM37';

% Start parallel computing
funList = {@CO2,@Ardu};
dataList = {data1,data2};

p = parpool(2);
spmd
    funList{labindex}(dataList{labindex});
end
delete(gcp)
PlotResults

function CO2(data1)
% unpack input
sdir     = data1.sdir;
CapnoCOM = data1.CapnoCOM;
Capnograph

function Ardu(data2)
% unpack input
sdir     = data2.sdir;
ArduCOM  = data2.ArduCOM;
ArduinoControl
```

Capnograph.m

```
%% predefine some parameters
ArduId         = 2;
```

```matlab
FinalData       = zeros(1,4);
lastBreathIdx   = 1;
BreathIdxs      = [];
time            = 0;
CO2SMOleft      = [];
PETCO2          = [];
Stop            = 0;
ExtraDeadSpace = [];
T_min           = 60;

%% Parameters to fill in
T_end           = 5;                  % End time in minutes
T_end           = T_end * 60;         % End time in seconds
PETCO2Ref       = 5.3;                % kPa
PDS             = (550-170)/(2^3); % ml/dkPa added dead space (ADS)
IDS             = 4;                  % ml/dkPa/Breath ADS
AntiHyperVentilation = IDS*(0.05^(1/3));  % ml ADS reduction per breath
IntegralDeadSpace = 280;             % Initial ADS for integration
                                      % This is the ADS untill the Smallest ADS
                                      % with closed valve

%% Make serial connection
load('CapnographCommands.mat')
s1 = serial(CapnoCOM);
s1.BaudRate = 19200;
s1.InputBufferSize = 200000;
s1.timeout = 160;
fopen(s1);

%% Initiation
fprintf(1,'Initializing CO2SMO plus!\n');
fwrite(s1, CommIn{1})
AnsIn1 = fread(s1,4,'uint8');
fwrite(s1, CommIn{2})
AnsIn2 = fread(s1,43,'uint8');

fwrite(s1, CommIdle{1})
AnsIdle1 = fread(s1,8,'uint8');
fwrite(s1, CommIdle{2})
AnsIdle2 = fread(s1,4,'uint8');
fwrite(s1, CommIdle{3})
AnsIdle3 = fread(s1,6,'uint8');

%% Prepare measurement
fprintf(1,'Preparing measurement\n');
fwrite(s1, CommIdle{1})
AnsIdle1 = fread(s1,8,'uint8');
fwrite(s1, CommStart{1})
AnsStart1 = fread(s1,16,'uint8');
fwrite(s1, CommStart{2})
AnsStart2 = fread(s1,588,'uint8');
fwrite(s1, CommStart{3})
AnsStart3 = fread(s1,112,'uint8');

%% Alert User
fprintf(1,'Start measurement\n');
fprintf(1,'in 3...');
pause(1)
fprintf(1,'2...');
pause(1)
```

```
fprintf(1,'1...\n');
pause(1)
fprintf(1,'now\n');

%% Start measurement
fwrite(s1, CommStart{4})
tic;

CO2SMO = fread(s1,5000,'int16');
t1 = toc;

CapnographSignals
while Stop == 0
    CO2SMO = fread(s1,1000,'int16');
    t1 = toc;
    if t1 > T_min
        fprintf(1,'Minuut verstreken\n');
        T_min = T_min+60;
    end
    CapnographSignals
    if labProbe(ArduId) == 1
        Stop = labReceive(ArduId, 201);
    end
end

%% save data
DataCapno.FinalData       = FinalData;
DataCapno.time            = time;
DataCapno.PETCO2          = PETCO2;
DataCapno.BreathIdxs      = BreathIdxs;
DataCapno.ExtraDeadSpace = ExtraDeadSpace;
DataCapno.PETCO2Ref       = PETCO2Ref;
save([sdir, 'DataCapno.mat'], 'DataCapno')

%% close capnograph
fwrite(s1, CommEnd{1})
AnsEnd1 = fread(s1,35,'uint8');

fclose(s1);
delete(s1)
clear('s1')
```

CapnographSignals.m

```
%% swap endian
CO2SMOswap = swapbytes(int16(CO2SMO(1:end)));
CO2SMOswap = [CO2SMOleft; CO2SMOswap];

%% isolate signals
lowValues = find(CO2SMOswap<-19000);

newData=zeros(length(lowValues)-1,4);
for i = 1:length(lowValues)-1
    newData(i,:) = CO2SMOswap(lowValues(i)+1:lowValues(i)+4)';
end

%% Save left over piece of data for next round
CO2SMOleft = CO2SMOswap(lowValues(end):end);
```

```
%% Remove gaps and offset
newData = (newData - 2^14) - ( floor( (newData - 2^14)./128 ).*64 );

%% Express in correct units and connect new data to the old data
% Flow (L/min)
newData(:,1) = newData(:,1).*0.1;
% Pressure (kPa)
newData(:,2) = newData(:,2)./75;
% Capnogram (kPa)
newData(:,3) = newData(:,3)./75;
% Inhaled Volume (ml)
% newData(:,4) = newData(:,4)

FinalData = cat(1,FinalData,newData);

%% Time (s)
ti = linspace( time(end) , t1 , size(newData,1) + 1 );
time  = [time, ti(2:end)];

%% VCap
CapnographVCap
```

CapnographVCap.m

```
%% prepare data.
Volume = FinalData(lastBreathIdx:end,4);
CO2    = FinalData(lastBreathIdx:end,3);
Flow   = diff(Volume);

%% Find the beginning and end of the expirations.
[VolumeHigh,IndHigh] = findpeaks(Volume);
[VolumeLow,IndLow] = findpeaks(-Volume);
if ~isempty(IndLow) && ~isempty(IndHigh)
    IndLow = IndLow - 2; % In case the exhaled volume is smaller than
                         % the inhaled volume
    % Only whole expirations are used.
    if IndHigh(1) > IndLow(1)
        IndLow(1) = [];
        VolumeLow(1) = [];
    end
end
if ~isempty(IndLow) && ~isempty(IndHigh)
    if IndLow(end) < IndHigh(end)
        IndHigh(end) = [];
        VolumeHigh(end) = [];
    end
end

%% calculate Tidal Volume
% and discard dead space ventilateion breaths
if ~isempty(IndLow) && ~isempty(VolumeHigh) && ...
        length(VolumeHigh) == length(VolumeLow)
    TidalVolume = VolumeHigh - Volume(IndLow);
    IndLow(TidalVolume<150) = [];
end

%% find the PETCO2
if ~isempty(IndLow)
```

```
    % Get the CO2 points
    PETCO2_i = zeros(size(IndLow));
    for ii = 1:length(IndLow)
        if ii == 1
        PETCO2_i(ii) = max(CO2(1:IndLow(1)));
        else
            PETCO2_i(ii) = max(CO2(IndLow(ii-1):IndLow(ii)));
        end
    end
    IndLow(PETCO2_i < 1)    = []; % Sometimes the measurements are
    PETCO2_i(PETCO2_i < 1) = []; % interupted, this deletes some of
                                 % those points.
end

%% send Data to the Arduino
if ~isempty(IndLow) && ~isempty(PETCO2_i)
    Capnograph2Arduino

    PETCO2 = [PETCO2; PETCO2_i];
    BreathIdxs    = [BreathIdxs; IndLow + lastBreathIdx - 1];
    lastBreathIdx = lastBreathIdx + IndLow(end) - 1;
end
```

Capnograph2Arduino.m

```
% Integral reacts more for small differences ^(1/3)
% Proportional reacts more for big differences ^3

ProportionalDeadSpace = PDS * (PETCO2Ref - PETCO2_i(end)).^3;
for jj = 1:length(PETCO2_i)
    if (PETCO2Ref - PETCO2_i(jj)) ~= 0
        IDS_term = IDS * (PETCO2Ref - PETCO2_i(jj)) / ...
            abs(PETCO2Ref - PETCO2_i(jj)) * ...
            abs((PETCO2Ref - PETCO2_i(jj))).^(1/3);
    else
        IDS_term = 0;
    end
    IntegralDeadSpace     = IntegralDeadSpace + IDS_term - ...
                        AntiHyperVentilation;
    CO2Data.EDS = ProportionalDeadSpace + IntegralDeadSpace;
    if CO2Data.EDS > 627%polynomial to calculate VProxRef goes up above 627
        CO2Data.EDS = 627;
    end
    ExtraDeadSpace = [ExtraDeadSpace; CO2Data.EDS];
    % don't let it go too low if the Reference for PETCO2 is set too low.
    IntegralDeadSpace(IntegralDeadSpace < 250) = 250;
end
if t1 > T_end
    Stop = 1;
end
CO2Data.Stop = Stop;
labSend(CO2Data, ArduId, 102)
```

ArduinoControl.m

```
CO2Id = 1;
Stop  = 0;
Ardu = arduino(ArduCOM, 'uno');
```

```
ArduServo = servo(Ardu, 4, 'MinPulseDuration', 1000*10^-6,...
    'MaxPulseDuration', 2200*10^-6);

DataArdu.VProxSen = [];

% Constant values
% Proximity Sensor
load('ProximityPolynomials.mat')    % Polynomial values for the estimation
                                    % of the reference Voltage for the
                                    % proximity sensor (4th degree)
VProxRefMax     = 1.75;             % Maximal Voltage for proximity sensor
                                    % (minimal volume).
VProxRefMin     = 0.36;             % Minimal Voltage for proximity sensor
                                    % (maximal volume).
VProxRefMiddle  = 0.8;              % Reference Voltage of proximity sensor
                                    % to put the tube in the middle
% Servo Motor
SerMotPosOpen   = 900/(2200-1000);  % Position of servomotor where extra
                                    % dead space is minimal
SerMotPosClosed = 100/(2200-1000);  % Position of servomotor where extra
                                    % dead space is variable
% DC Motor
DCMotDirBigger  = 0;                % The direction for the DC motor where
                                    % the dead space gets bigger.
DCMotDirSmaller = 1;                % The direction for the DC motor where
                                    % the dead space gets smaller.
DCMotOn         = 5;                % Voltage to turn speed of the DC motor
                                    % to max
DCMotOff        = 0;                % Voltage to turn speed of the DC motor
                                    % to min

writePosition(ArduServo, SerMotPosOpen)      % Turn Servo to Open
VProxRef  = VProxRefMax;
VProxSen0 = readVoltage(Ardu,0);
VProxSen1 = readVoltage(Ardu,0);

while labProbe(CO2Id) == 0 && Stop == 0     % Probe for data from CO2 sensor
    % in the mean time, move the DC-Motor to the smallest position.
    VProxSen2 = VProxSen1;
    VProxSen1 = VProxSen0;
    VProxSen0 = readVoltage(Ardu,0);
    VProxSen  = median([VProxSen0, VProxSen1, VProxSen2]);

    % Define the direction and speed of the DC-motor
    if VProxSen < VProxRef - 0.01      % Dead space is too big
        DCMotDir   = DCMotDirSmaller;
        DCMotSpeed = DCMotOn;
    elseif VProxSen > VProxRef + 0.01 % Dead space is too small
        DCMotDir   = DCMotDirBigger;
        DCMotSpeed = DCMotOn;
    else                              % Dead space is about right
        DCMotDir   = DCMotDirSmaller;
        DCMotSpeed = DCMotOff;
    end

    VStop = readVoltage(Ardu, 5);     % Read for stop command
    if VStop > 3
        Stop = 1;
        labSend(Stop,CO2Id,201);
    end
```

```matlab
    % Send signals to Arduino
    writeDigitalPin(Ardu, 2, DCMotDir)   % Direction of DC-motor
    writePWMVoltage(Ardu, 3, DCMotSpeed) % Speed of DC-Motor

end

CO2Data = labReceive(CO2Id,102);   % Receive data from the CO2 calculations
Stop = CO2Data.Stop;

while Stop == 0

    % Get data from the other CPUs
    if labProbe(CO2Id) == 1; % CO2 CPU
        CO2Data = labReceive(CO2Id,102);
        Stop = CO2Data.Stop;
    end

    % Set reference voltage for the proximity sensor.
    VProxRef = PolyProx(1)*CO2Data.EDS^4 + PolyProx(2)*CO2Data.EDS^3 + ...
        PolyProx(3)*CO2Data.EDS^2 + PolyProx(4)*CO2Data.EDS^1 +...
        PolyProx(5);


    % Read the voltage from the proximity sensor.
    VProxSen2 = VProxSen1;
    VProxSen1 = VProxSen0;
    VProxSen0 = readVoltage(Ardu,0);
    VProxSen  = median([VProxSen0, VProxSen1, VProxSen2]);

    % Define position of the servo motor
    if VProxRef < VProxRefMin
        VProxRef = VProxRefMin;
    end

    if VProxRef > VProxRefMax % The target dead space is smaller then the
                              % minimally allowed dead space
        SerMotPos = SerMotPosOpen;
        VProxRef  = VProxRefMax;
        DataArdu.VProxSen = [DataArdu.VProxSen, 0];
    else
        SerMotPos = SerMotPosClosed;
        DataArdu.VProxSen = [DataArdu.VProxSen, VProxSen];
    end

    % Define the direction and speed of the DC-motor
    if VProxSen < VProxRef - 0.01      % Dead space is too big
        DCMotDir   = DCMotDirSmaller;
        DCMotSpeed = DCMotOn;
    elseif VProxSen > VProxRef + 0.01 % Dead space is too small
        DCMotDir   = DCMotDirBigger;
        DCMotSpeed = DCMotOn;
    else                               % Dead space is about right
        DCMotDir   = DCMotDirSmaller;
        DCMotSpeed = DCMotOff;
    end

    % Send signals to Arduino
    writeDigitalPin(Ardu, 2, DCMotDir)   % Direction of DC-motor
    writePWMVoltage(Ardu, 3, DCMotSpeed) % Speed of DC-Motor
    writePosition(ArduServo, SerMotPos)

    VStop = readVoltage(Ardu, 5);        % Read for stop command
    if VStop > 3
        Stop = 1;
        labSend(Stop,CO2Id,201);
    end
end
DataArdu.ExtraDeadSpace = polyval(InvPolyProx, DataArdu.VProxSen);
writePWMVoltage(Ardu, 3, DCMotOff)       % Turn DC Motor off
writePosition(ArduServo, SerMotPosOpen)     % Turn Servo to Open
pause(2)
save([sdir, 'DataArdu.mat'], 'DataArdu')
```

PlotResults.m

```
% %% load data
load([sdir, 'DataArdu.mat'])
load([sdir, 'DataCapno.mat'])

%% Plot Capnograph Signals

figure(1);
% Flow
hh(1)=subplot(2,2,1);
plot(DataCapno.time,DataCapno.FinalData(:,1)), hold on
ylim([-150 150])
title('Air flow')
ylabel('Flow (L/min)')
xlabel('Time (s)')

% Pressure
hh(4)=subplot(2,2,4);
plot(DataCapno.time,DataCapno.FinalData(:,2))
ylim([-4.5 4.5])
title('Air pressure')
ylabel('Pressure (kPa)')
xlabel('Time (s)')

% PCO2
hh(2)=subplot(2,2,2);
plot(DataCapno.time,DataCapno.FinalData(:,3)), hold on
plot(DataCapno.time(DataCapno.BreathIdxs),DataCapno.PETCO2, 'or')
plot([DataCapno.time(1),DataCapno.time(end)], ...
    [DataCapno.PETCO2Ref, DataCapno.PETCO2Ref], 'g')
ylim([0 7])
title('Capnogram')
ylabel('P_C_O_2 (kPa)')
xlabel('Time (s)')
legend('PCO2', 'P_E_TCO2', 'P_E_TCO2_R_e_f')

% Inhaled Volume
hh(3)=subplot(2,2,3);
plot(DataCapno.time,DataCapno.FinalData(:,4))
ylim([-100 2000])
title('Inspired Volume')
ylabel('Volume (mL)')
xlabel('Time (s)')
```

```
linkaxes(hh,'x')

%% plot Extra dead space

figure(2)
plot(DataCapno.time(DataCapno.BreathIdxs),...
    DataCapno.ExtraDeadSpace), hold on
plot(linspace(0,DataCapno.time(DataCapno.BreathIdxs(end)),...
    length(DataArdu.ExtraDeadSpace)),DataArdu.ExtraDeadSpace,'r')
xlabel('Time (s)')
ylabel('Extra Dead Space (ml)')
title('Extra dead Space')
legend('Target Dead Space', 'Dead Space')
% ylim([0 700])
```