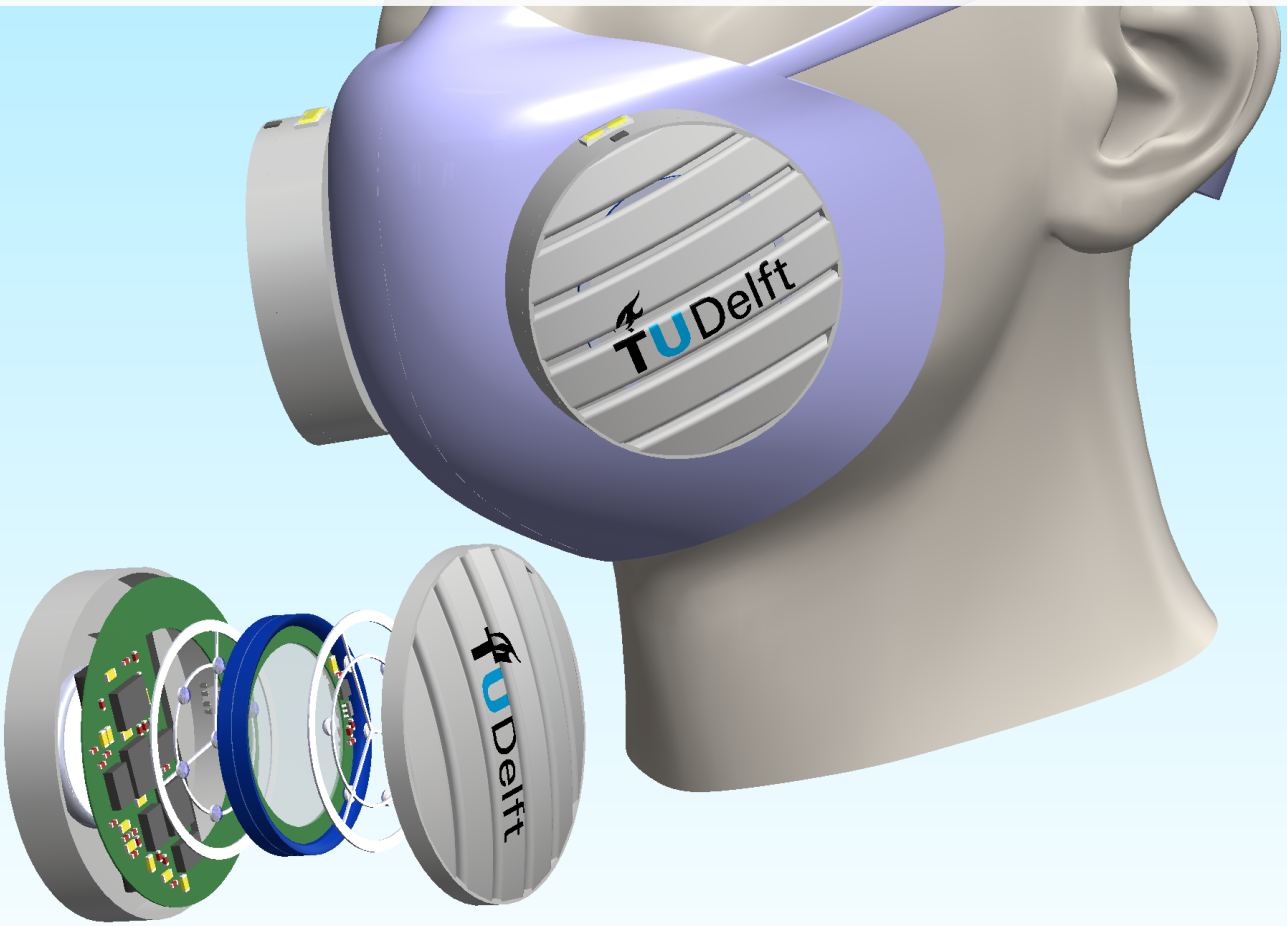
Smart personal protective equipment
the future of face masks

Sensing and Control

Michael Goddijn
Jasper-Jan Lut



TUDelft

Smart personal protective equipment
the future of face masks

Sensing and Control

by

Michael Goddijn
Jasper-Jan Lut

To obtain the degree of Bachelor of Science
at the Delft University of Technology,

Student numbers:	M. Goddijn,	4666968
	J. J. M. Lut	4698207
Project duration:	April 20, 2020 – July 3, 2020	
Supervisor:	Dr. Ing. H. W. van Zeijl	TU Delft
Thesis committee:	Prof. Dr. Ir. W. D. van Driel,	TU Delft
	Dr. Ing. H. W. van Zeijl,	TU Delft
	Dr. S.D. Cotofana,	TU Delft

In contribution with the entire SPPE group in the Introduction
up to and including Section 1.1.
BSc. Electrical Engineering 2020 BAP group C:

R. P. M. Bakker	4662482
M. J. H. Brouwers	4728181
H. J. Donkers	4475941
M. Goddijn	4666968
J. J. M. Lut	4698207
C. Zwart	4430867

TU Delft Delft University of Technology

Abstract

The current COVID-19 pandemic shows the necessity of personal protective equipment and face masks. In the project, a filter module with an in-situ ultraviolet-sterilization technique is designed that can serve as a new kind of *smart* personal protective equipment (SPPE). This technique is not used in wearable devices as of yet. The project thus aims to take the next step into the future of face masks.

The complete SPPE design is split into three submodules. In this thesis, the Sensing and Control submodule is designed.

The Sensing and Control submodule is divided into three parts as well. In the first part of the design, a negative feedback control loop is developed. A photodiode transimpedance amplifier circuit provides the feedback, and the controller is programmed on a microcontroller. The control parameters are derived from a model in Simulink. In the second part of the design, the temperature and relative humidity are measured to transform the control loop's reference value into a reference function. In the third part of the design, an estimation of the filter state of health is made by measuring the pressure drop over the filter material. Additionally, the airflow in the SPPE is calculated using equations from fluid mechanics to set the maximum allowable pressure drop.

At the end of the thesis, the Sensing and Control submodule allows the SPPE to measure environmental conditions, control the ultraviolet intensity accordingly, and indicate if the filter requires replacement. The design is finalized with printed circuit board designs and an algorithm. With the design of the Sensing and Control submodule, the next step is taken towards the future of face masks.

Preface

This document is the Sensing and Control thesis, which is part of the Smart Personal Protective Equipment (SPPE) graduation project. The graduation project forms the final fulfilment of the Bachelor of Science Electrical Engineering at the Delft University of Technology.

Right before the Graduation Project would start, the COVID-19 pandemic struck the world. At a minutes notice, we changed from our original project plans to the SPPE project. It is a project very relevant to society and the current needs for personal protective equipment.
The whole project has been a great learning experience. Even though no prototype could be build, we designed the Sensing and Control submodule as in-depth as possible, such that it can be produced with minimal extra research and resources. We hope that the project and this thesis contribute to a new generation of *smart* personal protective equipment.

We thank Dr. Ing. H.W. van Zeijl, our project proposer and supervisor, for his innovative ideas for the SPPE project, his guidance throughout the whole project, and the outstanding artist impressions of the SPPE that he provided us with.
We also thank our colleagues of the project group, Roy Bakker, Huub Donkers, Lotte Zwart, and Marcel Brouwers, for the fantastic and fun collaboration. Even though all our meetings took place online, the project is one never to be forgotten.
Further, we would like to show our gratitude towards Dr. Ing. I.E. Lager for the organisation of the bachelor graduation project. We especially appreciate the course organisation in these unique circumstances, and we are very grateful that the project organisation allows us to still graduate this academic year.

To conclude, we thank our families and friends for all the support they gave us throughout the past three years of our studies.

Michael Goddijn
Jasper-Jan Lut
Delft, June 2020

Glossary

CIPM	International Committee for Weights and Measures
COVID-19	Coronavirus disease 2019
DNA	Deoxyribonucleic acid
I^2C	Inter-integrated circuit communication protocol
LED	Light emitting diode
MERS	Middle East respiratory syndrome
MUX	Multiplexer
PCB	Printed circuit board
PPE	Personal protective equipment
PID	Proportional integral differential
PWM	Pulse-width modulation
RNA	Ribonucleic acid
SARS	Severe acute respiratory syndrome
SARS-CoV-2	SARS-coronavirus-2, the virus causing COVID-19
SPPE	Smart personal protective equipment
UV	Ultra violet
UVC	Ultra violet C ($\lambda \in [200, 280]$ nm)
UVGI	Ultra violet germicidal irradiation

$\mathbf{A}$	[m^2]	Area
$\mathbf{A_{eff}}$	[m^2]	Effective area
$\mathbf{C}$	[F]	Capacitance
$\mathbf{C_b}$	[F]	Bus capacative load
$\mathbf{C_d}$	[-]	Discharge coefficient
$\mathbf{C_{cycle}}$	[Ah]	Battery capacitance per cycle
$\mathbf{D}$	[J/m^2]	Dose
$\mathbf{D_{min}}$	[J/m^2]	Minimum dose
$\mathbf{e}$	[-]	Error signal
$\mathbf{E}$	[J]	Energy
$\mathbf{I}$	[W/m^2]	Intensity
$\mathbf{I_m}$	[W/m^2]	Measured intensity
$\mathbf{I_{max}}$	[W/m^2]	Maximum intensity
$\mathbf{I_{min}}$	[W/m^2]	Minimum intensity
$\mathbf{k}$	[m^2/J]	Inactivation constant
$\mathbf{k_0}$	[m^2/J]	Classical inactivation constant
$\mathbf{P_{abs}}$	[Pa]	Absolute pressure
$\mathbf{P_{dyn}}$	[Pa]	Dynamic pressure
$\mathbf{P_{in}}$	[Pa]	Pressure inside the SPPE
$\mathbf{P_{out}}$	[Pa]	Pressure outside the SPPE
$\mathbf{Q}$	[m^3/s]	Air flow
$\mathbf{r}$	[-]	Reference signal
$\mathbf{R}$	[Ω]	Resistance
$\mathbf{R_{p,max}}$	[Ω]	Maximum pull-up resistance
$\mathbf{R_{p,min}}$	[Ω]	Minimum pull-up resistance
$\mathbf{R_{spec}}$	[A/W]	Responsivity
$\mathbf{RH}$	[%]	Relative humidity
$\mathbf{S}$	[-]	Fraction of survival
$\mathbf{t}$	[s]	Time
$\mathbf{t_0}$	[s]	Radiation time
$\mathbf{t_{cycle}}$	[s]	Cycle time
$\mathbf{T}$	[K]	Temperature
$\mathbf{u}$	[-]	Controller output signal
$\mathbf{v}$	[m/s]	Velocity
$\mathbf{V}$	[V]	Voltage
$\mathbf{\Delta P}$	[Pa]	Differential pressure
$\boldsymbol{\lambda}$	[m]	Wavelength
$\boldsymbol{\lambda_{WPR}}$	[m]	Wavelength of peak responsivity
$\boldsymbol{\rho_{air}}$	[kg/m^3]	Air density
$\boldsymbol{\tau_{rise}}$	[s]	Rise time

Contents

1 | Introduction

The EE3L11 GRADUATION PROJECT forms the last course in the curriculum of the Bachelor Electrical Engineering of the Delft University of Technology. In the project, a group of six students investigates an electrical engineering challenge and develops a solution to it. Ideally, the solution is an electrical system, which is assembled into a prototype and tested by the students. However, due to the recent developments concerning the COVID-19 pandemic, the university decided that it is prohibited to build a prototype. Instead, the group must create its solution based on literature review and simulations.

The world is captured by the COVID-19 pandemic in the time that this project is conducted. The pandemic demonstrates the need for respiratory protection in health institutions to reduce the risk of infection for the workers. Particularly the shortage of face masks [1], generally designed for single-use, leads to the question if microelectronics could be applied to make air filters smarter in order to extend their use and improve their protection.

To design a smart, self-sterilizing air filter targeting a virus, the properties of that virus must be known. However, the virus that causes COVID-19 is a novel coronavirus, called SARS-CoV-2. As it is new to the scientific world, there is currently minimal data available. In this century, two other coronaviruses have caused epidemics. In 2003 the SARS-coronavirus was the cause of the *severe acute respiratory syndrome* (SARS) outbreak. From 2012 until the present day, the MERS-coronavirus circulated and caused the Middle East respiratory syndrome (MERS) [2].
The authors aim to provide the most relevant data in this thesis regarding the novel coronavirus. When we need to, we rely on information regarding the other coronaviruses, or viruses of different kinds. It will be indicated which virus is considered in given data.

Bacteria, viruses and other pathogens can be killed or inactivated by *ultra violet germicidal irradiation* (UVGI) [3]. Today, several techniques are available to irradiate pathogens with ultraviolet (UV) light in order to disinfect the air, surfaces, and drinking water. These techniques created applications such as air filtering systems with UV lighting, cabinets in which hospitals can sterilize their face masks, and continuous overhead UV lighting in laboratories [3]. However, none of these applications contains a small and mobile application of UVGI.

It is important to emphasize that the technique of UVGI sterilization can be implemented in different form factors, such as stationary applications where sterile air is crucial. Here, UVGI sources are installed as a filter module in an air conduct. These devices are beneficial in the medical and food industry. Other domains of interest are for example military as a form of protection or water filtration systems to remove any unwanted pathogens.

For this graduation project the group proposes the use of Smart Personal Protective Equipment, SPPE in short, in order to increase the effectivity and lifetime of face masks. We propose the SPPE as a filter module with an in-situ disinfection functionality based on UVGI. The key to the SPPE is the integration of a UV source in the filter combined with sensors to monitor the sterilisation process and filter performance. Two SPPE modules are connected to a face mask to create the functioning product: a smart face mask.

1.1 The SPPE project overview

In this section, an overview of the entire project is presented. It will also be discussed how the project is divided, what the interconnections are, and what the parameters are of each design parts. For each part, a separate thesis is written with this section as a general introduction. In the following chapters of this thesis, a detailed explanation is given on the work done to tackle the particular problems of the thesis group.

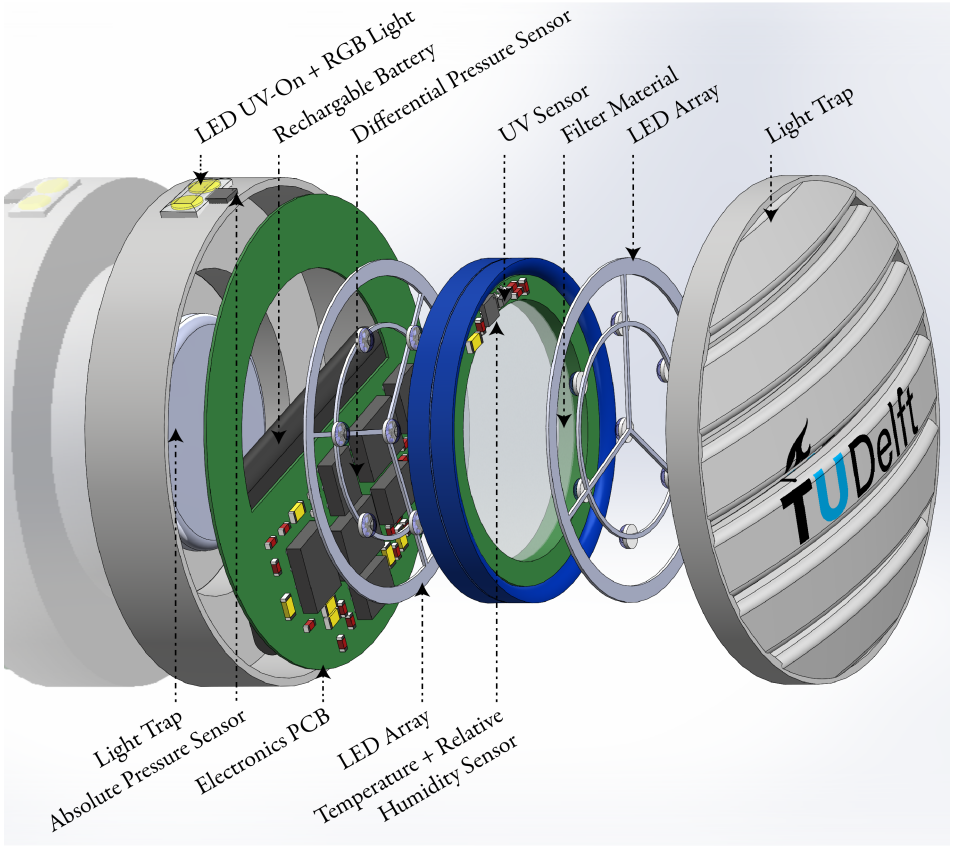


Figure 1.1: Exploded view of the SPPE. It includes the two air traps, the electronics layer, two LED arrays, and the filter material. The layers are combined with integrated pressure, relative humidity, temperature and UV sensors.

1.1.1 Design scope

The final design of the SPPE consists of two smart filter modules which are applied to a face mask. Each module can operate autonomously and contains space for replaceable filter material. When a SPPE module is operational, the filter is periodically irradiated with UV light to neutralize pathogens that are caught in the filter material. Sensors measure the air temperature, relative humidity, and the pressure to compute the required irradiation intensity and to monitor the quality of the filter material. When the filter material has reached the end of its life, the SPPE will notify the user to have it replaced. The SPPE is battery powered. It can operate for eight hours before needing to be recharged. Figure 1.1 shows the filter module's components implemented in a exploded view.
A physical demonstrator of the SPPE is not realized, due to the COVID-19 restraints. Figures 1.2 and 1.3 give an impression of what the SPPE would look like.

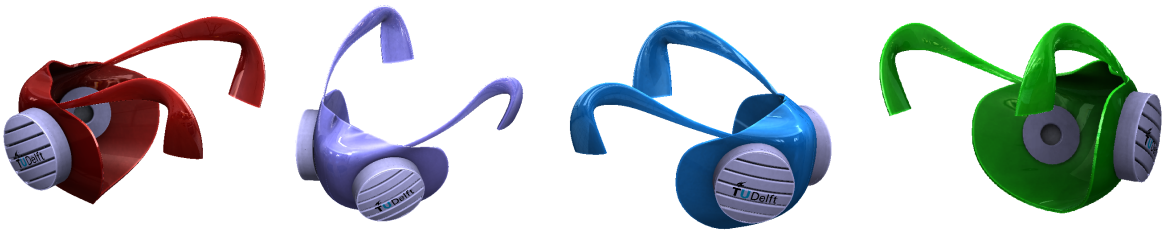
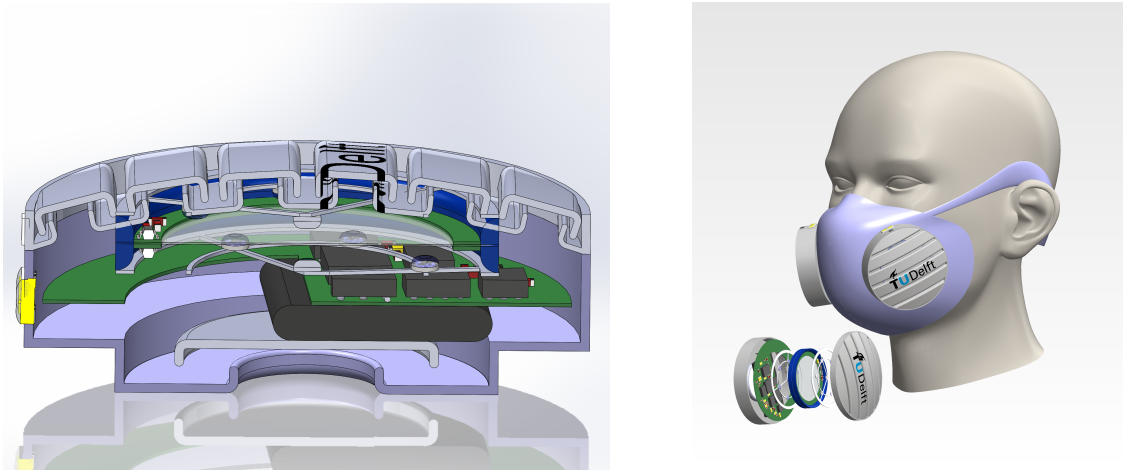


Figure 1.2: Full view of the SPPE.



(a) Cross-sectional view of the SPPE.

(b) Two SPPE modules as connected on a face mask.

Figure 1.3: Artist impressions of the SPPE.

1.1.2 Division in submodules

The project group is divided into three teams, consisting of two students each. The teams are organised in the following manner:

Ultraviolet Germicidal Irradiation (UVGI)	R.P.M. Bakker
	M.J.H. Brouwers
Sensing and Control (SaC)	M. Goddijn
	J.J.M. Lut
On-Board Power Management (OBPM)	H.J. Donkers
	C. Zwart

Each team tackles part of the SPPE design. These parts are referred to as a *submodules*. A brief introduction to every submodule follows below.

Ultraviolet Germicidal Irradiation

The Ultraviolet Germicidal Irradiation submodule is responsible for providing the UV radiation in order to disinfect the filter material in the SPPE. This is achieved through the use of LEDs radiating in the UVC spectrum (wavelengths between 200 and 280 nm). The UVGI submodule selects LEDs with the most effective wavelength, and ensures the most optimal placement of the UVC LEDs to achieve uniform irradiation, with accuracy and power consumption in mind. Additionally, the UVGI submodule will be design the UVC LED drivers. These drivers are controlled by a signal provided by the SaC submodule.

Sensing and Control

The Sensing and Control submodule is the controlling body of the SPPE. The submodule measures the temperature, relative humidity, air pressure, and the pressure drop over the filter material. It generates the control signal for the UVGI submodule, thereby regulating the intensity of the UVC radiation. The Sensing and Control submodule ensures a safe (always sufficient UV intensity) and power-efficient (not an unnecessarily high UV intensity) operation of the SPPE by using a control loop. It also monitors the quality of the filter material.

On-Board Power Management

The On-Board Power Management submodule aims to provide energy for the sensor circuitry and the LED modules. Since the SPPE is a wearable device, the system needs to have on-board power with an energy management system for low-power application and battery safety. The on-board power is implemented by making use of a rechargeable battery and a battery management system that controls and protects the battery. The goal is to design a system that can supply the SPPE with a specified power level safely and efficiently.

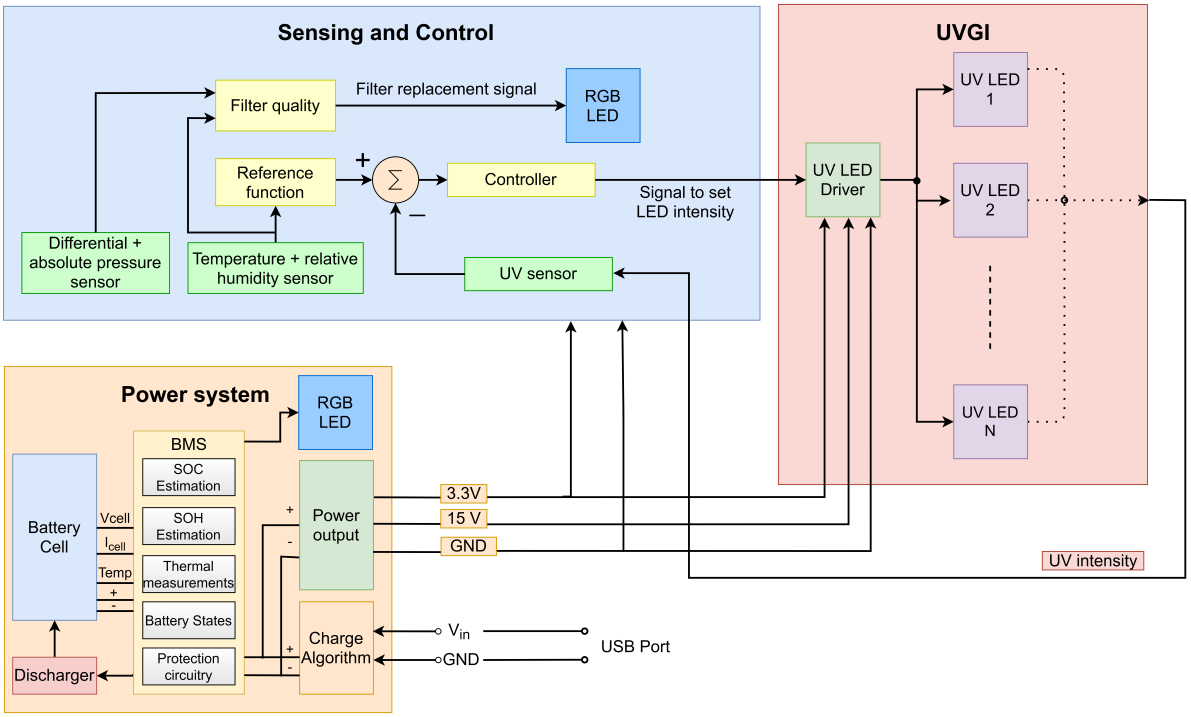


Figure 1.4: The SPPE system divided into submodules.

1.1.3 Submodule interconnections

The submodules are interconnected physically and through design constraints. Figure 1.4 shows the three physical interconnections, which are: the power outputs from the OBPM submodule to the other submodules, the control signal for the UVC LEDs from the SaC module, and the feedback of the UV intensity to the SaC module.
The submodules of the SPPE are closely related, such that design constraints also connect them. For example, the diameter of the filter material influences the number of UVC LEDs needed to radiate the entire filter with an adequate dose. Using more UVC LEDs influences how the control loop is calibrated. It also means that more power and higher peak currents are necessary, which influences the battery size. As the battery is the largest component, this has an influence on the filter diameter and the weight of the filter module itself. The complete list of design constraints is listed in the next chapter.

These interconnections illustrate the need for communication between the subgroups about these parameters during the design process. Communication is vital to ensure an optimal SPPE system design.

1.2 Problem definition

In order to ensure the safety of the person wearing a face mask with SPPE modules (referred to as *the user*), the intensity must be high enough to inactivate the viruses in a limited amount of time. On the other hand, the UV intensity must not consume power unnecessarily, as the SPPE is a battery-powered system.
As mentioned, there exists no wearable or *smart* personal protective equipment yet. The SPPE is thus considered as a *technology demonstrator*, i.e. it shows what is possible in this unexplored field of products.

Based on these considerations, our thesis will focus on answering the following questions:

1. How can we ensure the correct UV dose is applied to the filter material?
2. How can we use sensors near the filter material to adjust the UV intensity, hereby controlling the UVC LEDs in a power-efficient way?
3. How can microelectronics be used in other ways to put the *smart* in SPPE?

1.3 State of the art

What are the state-of-the-art technologies in the field of UVGI, sensors, and control systems?

As mentioned already, there are products available that use UVGI to sterilize medical equipment. Some examples of this are UVGI cabinets [4] and UVGI room cleaners [5]. The most mobile products are cases used for sterilizing face masks, for example, by Spectroline [6]. However, none of these products concern wearable devices. Hence, as of yet, the SPPE is a one-of-a-kind concept.

One of the reasons that there are no wearable UVGI products yet is the recent development of new UV sources. Classically, the UV source used for UVGI is a mercury lamp, which radiates typically at 254 nm. Mercury lamps are large, heavy, and expensive [3]. During the last few years, more research has been conducted in Light Emitting Diodes (LEDs) that radiate in the UVC spectrum. These have a minimal form factor and can radiate at wavelengths that are closer to the optimal wavelengths of UVGI [3]. Several recent studies indicate that UVC LEDs can also be used in UVGI applications [7], [8], [9].

Several off-the-shelf sensors are used in this thesis. These are sensors to measure the UV intensity, temperature, relative humidity, absolute pressure, and differential pressure. Why and how these sensors are used will be explained in the later chapters. Instead of using off-the-shelf technology, many exciting developments are happening regarding flexible and ultra-small form factor sensors. However, as the SPPE is also a technology demonstrator, implementing these novel sensors is outside the scope of this thesis.
There is much research conducted on wearable UV sensors. In the coming years, it is expected that reliable wearable UV detectors will become available [10]. The recent development on wearable temperature sensors concerns flexible graphene sensors that remain accurate after many bending cycles [11], [12]. These sensors could be integrated into the face mask. Wearable humidity sensors are also being developed [13]. These are even able to detect breathing patterns based on capacitive changes [14]. Lastly, flexible graphene pressure sensors are also in development. These can also be used in respiratory applications or integrated in the face mask [15].

An insight regarding the stability control of the SPPE is considered to conclude this section. In the SPPE, a PID controller is implemented to ensure the stability of the UV intensity. However, a PID controller is tuned with a fixed set of parameters. This fixation can cause problems if components in the system start to drift. The technique of Adaptive PID control can be a solution to this problem. In this technique, a controller can adjust its values if it is deemed necessary [16].

1.4 Thesis outline

The Programme of Requirements is presented in the following chapter. The requirements are used throughout the thesis to make decisions for the Sensing and Control design. In Chapter 3, the controlling mechanism of the UV intensity is discussed, designed, and implemented. In Chapter 4, the effect of environmental conditions on pathogens is discussed, and the power-efficient control of the UV intensity is designed and implemented. In Chapter 5 a new functionality of the Sensing and Control submodule, safeguarding the filter quality, is discussed and implemented. In Chapter 6, the design of the submodule is finalized by creating in-depth software and hardware designs. Also, some safety considerations are listed, and it is discussed what is still needed to transform the submodule from a design into reality.
At the end of the thesis, a conclusion and a discussion are given. Here, recommendations are given on what to change, a possible reduction in cost and how to add more functionalities into the Sensing and Control submodule.

2 | Programme of Requirements

As described in the Introduction, the Sensing and Control submodule is one of the three parts of the SPPE. Because of how the project is set up, the Programme of Requirements is split into two sections: one listing the requirements for the whole SPPE product, and one listing the requirements that are specific for the Sensing and Control submodule.

2.1 Requirements for the SPPE

The goal of the SPPE project is to design a new kind of *smart* personal protective equipment that includes an in-situ disinfection functionality based on UVGI. The SPPE is proposed as a filter module, of which two can be connected to a face mask to create a user-protecting product.
In order to design a functioning product, some mandatory (M) requirements are set upon the design of the SPPE. These are as follows:

(M.I) The SPPE uses UVGI to sterilize the filter material.

(M.II) The SPPE form factor is small enough to fit on one side of a regular face mask.

(M.III) The SPPE's sides are airtight.

(M.IV) The SPPE's filter material is replaceable.

(M.V) The SPPE's electronics are reusable after the filter material has been replaced.

(M.VI) The SPPE's battery life is at least eight hours.

(M.VII) The SPPE's components lifetimes are at least two years.

(M.VIII) The SPPE must safeguard the user during operation. This requirement includes that no UV light reaches the user and that no harmful pathogens reach the user.

(M.IX) The SPPE should communicate internal system conditions to the user.

Design aspects
Besides the mandatory requirements for the SPPE, we define several non-mandatory requirements that are favourable to utilize. These requirements are also referred to as *design aspects* (D). The design aspects of the SPPE are as follows:

(D.I) The SPPE is preferably designed durably: if one electrical component breaks down, it can be replaced with replacing as little other components as possible.

(D.II) The SPPE is preferably designed using off-the-shelf electrical components.

(D.III) The SPPE is preferably constructed from light materials.

(D.IV) The SPPE should be designed power efficiently.

(D.V) The SPPE is designed as detailed as possible without manufacturing prototypes.

2.2 Requirements for the Sensing and Control submodule

The Sensing and Control submodule must control the UV intensity by measuring it and adjusting it to the environmental conditions. It also must monitor the filter quality and indicate when the filter material needs to be replaced. Besides the general requirements of the SPPE as listed above, several requirements are explicitly applicable to the Sensing and Control submodule. These are specifications of the functionalities of the Sensing and Control submodule. The requirements are as follows:

(M.i) The Sensing and Control submodule measures the UV intensity in the SPPE during the period of irradiation.

(M.ii) The Sensing and Control submodule compares the measured UV intensity with a reference value.

(M.iii) The Sensing and Control submodule sends a signal to the UVGI submodule during radiation. The signal will indicate the intensity that the UVC LEDs need to provide at the filter material.

(M.iv) The Sensing and Control submodule determines the reference value from measurements of the temperature, the relative humidity and a predefined base value.

(M.v) The Sensing and Control submodule radiates at least once every half hour. The UVGI team provided the following radiation values:
The radiation time $t_0 = 105$ seconds. The radiation wavelength $\lambda \in [275,\ 285]$ nm. The radiation intensity $I \in [0,\ 1.1]\,\frac{\text{mW}}{\text{cm}^2}$ at the filter material.

(M.vi) The Sensing and Control submodule measures the pressure drop over the SPPE's mechanical filter at least every half hour.

(M.vii) The Sensing and Control submodule compares pressure drop with a threshold that indicates if the filter is clogged and needs to be replaced. This threshold is derived from the European Union guideline for pressure drops across face masks.

(M.viii) The Sensing and Control submodule notifies the user if the filter material is clogged via an RGB LED on the outside of the SPPE.

(M.ix) The Sensing and Control submodule provides UVGI control on both sides of the filter material in the SPPE module (on the outside and on the inside).

(M.x) The Sensing and Control submodule's main printed circuit board must be smaller than 50x50 mm.

(M.xi) The Sensing and Control submodule's voltage supply is 3.3 V.

Design aspects

Besides the mandatory requirements for the Sensing and Control submodule, we define several design aspects (non-mandatory requirements) that are favourable to utilize. These are as follows:

(D.i) The Sensing and Control submodule is preferably designed to minimize its power consumption.

(D.ii) The Sensing and Control submodule is preferably designed as detailed as possible under the circumstances of the current pandemic which prohibit prototyping.

(D.iii) The Sensing and Control submodule is preferably designed such that it is capable of inactivating other kinds of pathogens with changing little to the submodule.

(D.iv) The Sensing and Control submodule is preferably designed such that it is applicable in some other UVGI device, either mobile or stationary, with changing little to the submodule.

3 | UVGI Control

The SPPE contains arrays of UVC LEDs that radiate onto the filter material. The radiation intensity at the filter material needs to be controlled for optimal filter performance and protection against harmful pathogens.

The following questions are answered in this chapter:

How does UVGI work?
How can a feedback mechanism be used to determine the UV intensity at the filter material?
How can the feedback be used to control the radiation at the filter?
How is this control implemented?

First, in Section 3.1, the working of UVGI is explained. Second, in Section 3.2, the feedback mechanism is designed. Third, in Section 3.3, the control mechanism is designed. Last, in Section 3.4, a microcontroller is selected for the implementation of control.

3.1 Principle of UVGI

First, a brief introduction to UVGI is given, the central mechanism of the SPPE project. Please note that this information is not crucial to understand the remainder of this thesis. Readers who assume UVGI as a given technology can skip Subsection 3.1.1. Here, we answer the following question for the interested readers: how does UVGI inactivate pathogens?

3.1.1 Effect of UV light on genetic material

Several types of viruses are distinguished, identified by the way their genetic material is structured: ssRNA, ssDNA, dsRNA and dsDNA, where ss stands for *single-stranded* and ds stands for *double-stranded*. The novel coronavirus SARS-CoV-2 is of the ssRNA type. The ssRNA-type virus requires the least amount of UV irradiation to get inactivated [17].
DNA (Deoxyribonucleic acid) is a large molecule made out of subunits called nucleotides. The nucleotide consists of three parts: deoxyribose, phosphate, and a nucleic acid-base. This acid-base is either thymine (T), adenine (A), cytosine (C), or guanine (G). The genetic information is encoded in the DNA using these acid-bases. RNA (ribonucleic acid) is structured likewise, except that uracil (U) takes the place of thymine. The acid bases of DNA always connect in the same pairs of T-A and C-G. The acid bases of RNA always connect in the same pairs of U-A and C-G.

Incident UV light causes cross-links between adjacent acid bases, forming so-called dimers, which cause mutations. These dimers cause the DNA/RNA to become "unreadable", and the cell is not able to reproduce the genetic information. Inactivation of the pathogen has now occurred. The formation of dimers is illustrated in Figure 3.1. Often, thymine-dimers are formed, as the T-A connection is weaker than the C-G connection (only two hydrogen bonds instead of three) ([3], Sec. 2.2).

The UV spectrum (200-400 nm) is separated into three parts: UVA (320-400 nm), UVB (280-320 nm), and UVC (200-280 nm). The Planck-Einstein relation explains that the energy of a photon is inversely proportional to its wavelength, i.e. $E \propto \lambda^{-1}$. The energy level required to form dimers in the DNA/RNA is relatively high. Hence, UVB or UVC light is used in UVGI applications. UVC light has also been proven to be very efficient in the inactivation of the SARS-CoV [18].

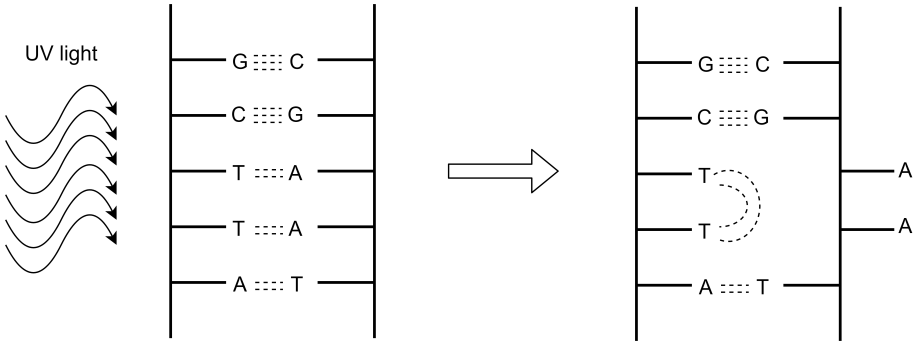


Figure 3.1: Dimer formation in a double-stranded DNA molecule under the influence of incident UV light. Note the number of hydrogen bonds (dotted lines) in the T-A connection opposed to the C-G connection. Source: M. Goddijn, adapted from ([3], p. 22).

3.1.2 UVGI quantities

Throughout this thesis, several quantities are considered that are related to UVGI. It is essential to note the difference between two of these: the *intensity* of the UV light and the *dose*.

- The intensity of the UV light is the amount of power radiated per unit area. It is denoted by I and is measured in $\frac{\mathrm{W}}{\mathrm{m}^2}$ or in $\frac{\mathrm{mW}}{\mathrm{cm}^2}$.
- The dose is the amount of energy that is incident on a unit area. It is denoted by D and is measured in $\frac{\mathrm{J}}{\mathrm{m}^2}$ or in $\frac{\mathrm{mJ}}{\mathrm{cm}^2}$.

The relation between I and D is given by Equation (3.1), here t denotes the time (in seconds) that the UV light is incident on the surface.

$$D = I \cdot t \tag{3.1}$$

3.2 UV sensor

The Sensing and Control submodule must measure the UV intensity, as is stated in Requirement (M.i). This is done by implementing a UV sensor in the SPPE. The goal of the UV sensor is to detect the radiated UV intensity at the filter material and to a signal this value to a controller. This signal can be communicated in many forms, such as Pulse Width Modulation (PWM), using digital $\mathrm{I}^2\mathrm{C}$ communication or by an analogue voltage, depending on what the controller can interpret. To be able to create a proper design, the following sensor requirements are proposed:

- The sensor needs to be able to detect an intensity of 0 to 1.1 $\frac{\mathrm{mW}}{\mathrm{cm}^2}$, as given by Requirement (M.v).
- The sensor needs to be sensitive to UV radiation with a wavelength $\lambda \in [275, 285]$ nm, as given by Requirement (M.v).
- The sensor output voltage range has to be between 0 and 3.3 V.

For now, it can be assumed that the chosen controller can interpret an analogue voltage between 0 and 3.3 V as the sensor output, more on this will follow in Section 3.4.

3.2.1 Sensor choice

In this section, a sensor is selected for translating the UV intensity to the electrical domain. A method of detection can be chosen that is suited for the SPPE based on the sensor requirements set above.

Several UV sensors are investigated. All of these translate the optical domain to the electrical domain using a photodiode. Photodiodes have a linear relation between the light intensity and its output current, which is a useful property for the SPPE ([19], pp. 635-639).
All photodiodes respond to a certain band of wavelengths, referred to as the *detection band*. The detection band is mostly between 230-280 nm for UVC photodiodes and between 230-310 nm for UVB photodiodes. Usually, the photodiode's responsivity, i.e. the current output for the light power input, varies within the detection band. The wavelength at which the peak response occurs is the *wavelength of peak responsivity*

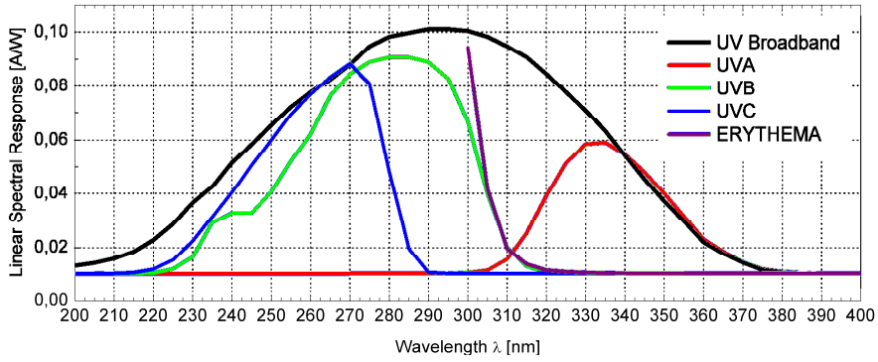
Table 3.1: Product information of the GUVB-S11SD by Roithner Lasertechnik [22].

Photodiode characteristic	**GUVB-S11SD**
Material	AlGaN
Diode type	Schottky diode
Dimensions (l, w, h)	2.8, 3.5, 1.9 mm
Forward current (max.)	1 mA
Reverse voltage (max.)	3 V
Operating temperature	[-30, +85] °C
Detection band	[235, 320] nm
Responsivity ($\lambda = \lambda_{WPR} = 300$ nm)	0.14 A/W
Photo current $\left(\text{at 306 nm, at 1 } \frac{\text{mW}}{\text{cm}^2}\right)$	69 nA

(λ_{WPR}). Figure 3.2a provides the responsivity curves of the different photodiodes, respective of their UV type. The sensor most suitable for the SPPE has its λ_{WPR} at 280 nm. From the figure, it turns out that this corresponds with a UVB photodiode. So, even though UVC light is detected, it will be best to consider UVB photodiode here.

Several semiconductor materials can be used to construct a photodiode. Some examples are silicon carbide (SiC), gallium phosphide (GaP), aluminium gallium nitride (AlGaN), and many more [19].
Most of the commercially available photodiodes are based on SiC, but this makes them very expensive to use in the SPPE (€50-€150 each). According to Blank *et al.* [20], SiC photodiodes are challenging to fabricate and are used in high-temperature and high-voltage applications. Kalinina *et al.* [21] emphasize the usage of SiC photodiodes in applications where the temperature is above 150 °C and the photons have an energy higher than 20 eV, i.e. the wavelength $\lambda < 62$ nm. Under less harsh conditions, other types of photodiodes can be used [21].
The SPPE will not be used under these harsh conditions, as stated in Requirement (M.v). Hence, the photodiode of the SPPE does not have to be of the expensive SiC type.

In Appendix A.1 a comparison is given between several photodiodes. The SPPE will use the GUVB-S11SD by Roithner Lasertechnik [22]. The diode is selected based on its good characteristics, and its low price (€4 ± €1). The GUVB-S11SD is based on a Schottky diode of the AlGaN type. According to Li *et al.* [23], the AlGaN photodiodes can be used in medical applications just as the expensive SiC photodiodes.
Relevant information about the GUVB-S11SD is summarized in Table 3.1. Figure 3.2b shows an image of the GUVB-S11SD. At the time of writing no data is published yet about the estimated lifetime of the GUVB-S11D, which should be known for Requirement (M.VII).



(a) The spectral responsivity of different types of UV photodiodes to individual wavelengths [24].

(b) The GUVB-S11SD UVB photodiode by Roithner Lasertechnik, its dimensions are (lxwxh): 3.5 x 2.8 x 1.9 mm [22].

Figure 3.2: Phododiode responsivity and the GUVB-S11SD UVB photodiode.

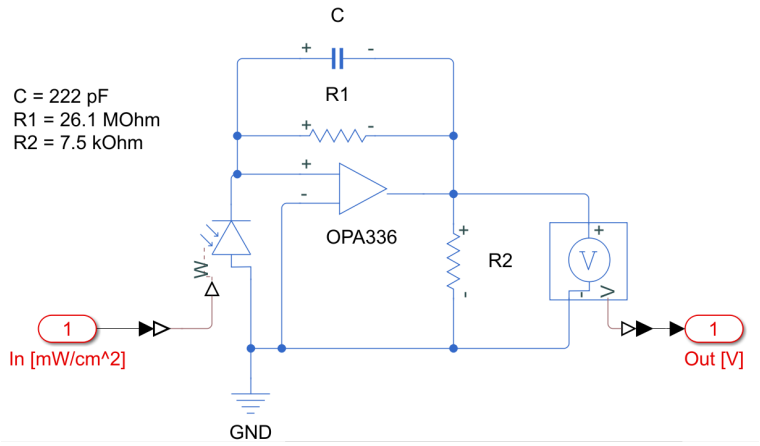


Figure 3.3: The transimpedance amplifier circuit using the OPA336 amplifier and GUVB-S11SD photodiode.

3.2.2 Sensing circuit

The photodiode outputs a current that is linearly proportional to the incident UV intensity. In this section, the output current is converted into an analogue voltage, such that the controller can interpret the signal.
A current can be converted into a voltage using a transimpedance amplifier ([25], Sec. 8.4). The design is based on the circuit given in the photodiode-selection guide of Sglux (manufacturer of UV photodiodes) [24] and on a transimpedance amplifier circuit given in the book on Structured Electronic Design ([25], Sec. 8.4). The circuit is shown in Figure 3.3. The resistor value (R) and capacitor value (C) are chosen such that the output voltage is in the range of [0, 3.3] V.

Equation (3.2) is used to determine the resistor value R [24]. Here, V_{cc} is the supply voltage of 3.3 V (Requirement (M.xi)), and Current_{max} the expected current output of the photodiode at maximum intensity.
The maximum current is determined by Equation (3.3). Here I_{max} is the maximum UV intensity (10.45 W/m^2), A_{eff} the effective area of the photodiode ($5.31 \cdot 10^{-8}$m^2), and $R_{spec}(\lambda = 280\text{nm})$ is the responsivity at 280 nm (0.13 A/W) [22].
Equation (3.4) is used to determine the capacitor value C [24]. Here, R is the resistance of Equation (3.2) and τ_{rise} is the desired rise time on the range of [0.01, 0.1] s. A fast rise time is desired for the UV sensor, hence we select $\tau_{rise} = 0.01$ s.

$$R = \frac{V_{cc} - 50 \cdot 10^{-3}}{\text{Current}_{max}} = \frac{3.3 - 50 \cdot 10^{-3}}{72.1 \cdot 10^{-9}} = 45.1\ \Omega \tag{3.2}$$

$$\text{Current}_{max} = I_{max} \cdot A_{eff} \cdot R_{spec}(\lambda = 280\text{ nm}) = 10.45 \cdot 5.31 \cdot 10^{-8} \cdot 0.13 = 72.1\text{ nA} \tag{3.3}$$

$$C = \frac{\tau_{rise}}{R} = \frac{0.01}{45.1 \cdot 10^{6}} = 222\text{ pF} \tag{3.4}$$

3.2.3 Sensor placement and implementation.

The UV sensor needs to be positioned in a location such that it can accurately determine the UV intensity at the filter material. This position is co-designed with the UVGI team. A outer ring around the filter material is designed that, among others, act as a support for the UV sensor, please refer to Figure 1.1. This outer ring is also to ensure that no part of the filter material is covered by the sensor, thereby preventing any shading of the filter material from the UV light. The orientation of the photodiode on the outer ring is such that it faces the UVC LEDs.

The UV sensor is implemented as shown in its circuit schematic of Figure 3.4. Here, the transimpedance amplifier circuit from Figure 3.3 is implemented using the OPA336 (Texas Instruments) [26]. This operational amplifier is recommended by the photodiode-selection guide as a low cost amplifier that suffices for this application [24].

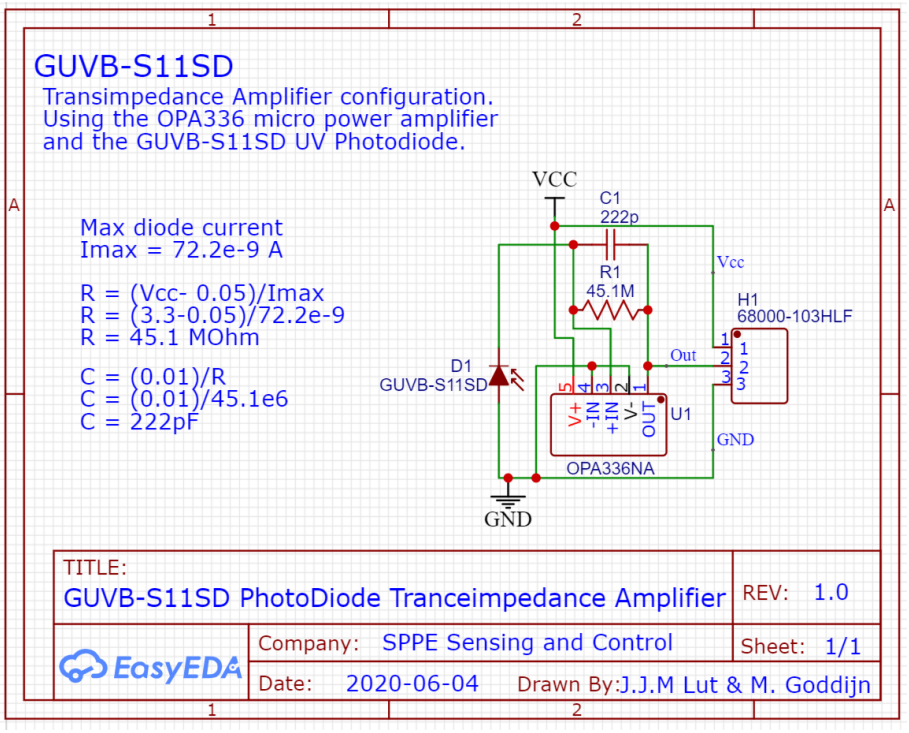


Figure 3.4: The circuit schematic of the UV sensor.

3.3 Control loop

The UV sensor's feedback is used in a control loop as negative feedback to determine the error in the measured intensity. The controller then regulates the intensity of the UVC LEDs based on this error.
The control loop is shown in Figure 3.5 in its most simple form. Here, $I(t)$ is the UV intensity radiated by the UVC LEDs onto the filter material. $I_m(t)$ is the intensity that is measured by the UV sensor. $r(t)$ is the reference representing the desired intensity, with which $I_m(t)$ is compared. The reference value is set beforehand. $e(t)$ is the error signal, with $e(t) = r(t) - I_m(t)$. The controller block contains some algorithm to translate $e(t)$ to a value $u(t)$ that is the input of the UVC LEDs. In the UVC LEDs, $u(t)$ is mapped to a new $I(t)$ by means of a current through them.

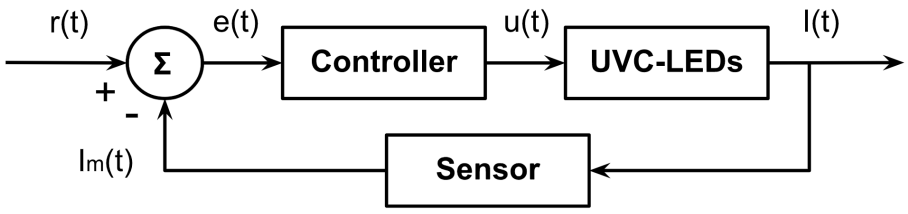


Figure 3.5: The negative feedback control loop.

3.3.1 Control theory basics

There are several ways of designing the controller. Three common relations exist between $e(t)$ and $u(t)$ that can be used to design the controller ([27], Ch. 4). These are:

- **Proportional control (P)**
  The proportional controller describes a linear relation between the $e(t)$ and $u(t)$: $u(t) = K_p e(t)$. The downside of this is that in some control systems, the signal tends to settle at a steady-state offset lower than the desired value. In other words, a steady-state error may exist using proportional control.

- **Integral control (I)**
  The integral controller looks at a past accumulation of $e(t)$: $u(t) = K_i \int_0^t e(\tau)\mathrm{d}\tau$. The goal of the integral controller is to compensate for the steady-state error as much as possible. The downside is that the integral controller can cause overshoot of the desired value and cause the system to oscillate around this steady-state value.

- **Differential control (D)**
  The differential controller makes a future prediction of the system and adjusts the rate of change accordingly: $u(t) = K_d \frac{\mathrm{d}e(t)}{\mathrm{d}t}$. This can compensate for the oscillatory behaviour of the integral controller.

For systems of at least the second order, a stable controller is obtained by using a combination of the three controllers listed above ([27], p. 222). In this way, the method of Proportional-Integral-Differential control (PID) is obtained. The controller equation is written as:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\mathrm{d}\tau + K_d \frac{\mathrm{d}e(t)}{\mathrm{d}t} \tag{3.5}$$

As of yet, the order of the system of Figure 3.5 is not known. Hence, the method of PID control is selected to ensure a stable operation.

3.3.2 Control loop design and validation

The negative feedback controller from Figure 3.5 is designed and implemented in the Simulink environment of MATLAB [28], as is shown in Figure 3.6. In this implementation, the PID controller is implemented as in Equation (3.5). The UVC LED system is provided by the UVGI team and the UV sensor is the circuit presented in Figure 3.3. The control loop aims to set $I(t)$ proportional to the reference value, which is assumed to be constant for now. More on the reference value follows in the Chapter 4.

The resulting output intensity of the UVC LEDs has to uphold to the following requirements, to verify if the control loop is implemented successfully:

- The control loop has a rise time of no more than 10% of the total irradiation time t_0.
- No overshoot may be present in the output signal as this could damage the UVC LEDs.
- A steady state error of 0% has to be achieved in a stable output signal.
- The design is insensitive to noise and other disturbances.

The rise time is defined as the time the signal takes to go from 10% to 90% of the desired value. This restriction is given as a fraction instead of some time in seconds, as the irradiation time t_0 can vary drastically, depending on the pathogen that is being inactivated. From the book of Kowalsky ([3], App. B) it is known that some viruses need a dose of only 0.1 mJ/cm^2. With the maximum intensity of 1.1 mW/cm^2, the irradiation time t_0 is one second. The maximum allowable rise time of the control loop is then 0.1 second.

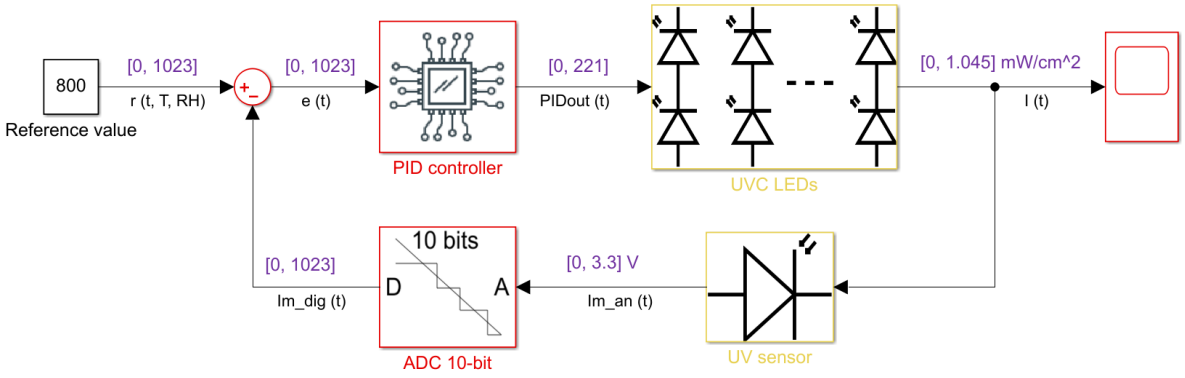
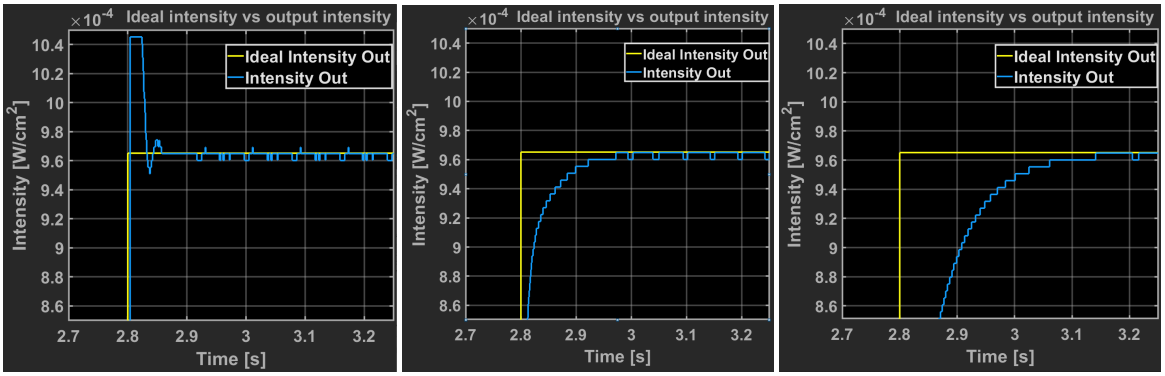


Figure 3.6: The control loop as it is implemented in Simulink. The ranges of the signal values indicated in purple, including the unit, if applicable.



(a) Overdamped response. $P = 0.65$, $I = 37.28$, $D = 0.001$

(b) Critically damped response. $P = 0.23$, $I = 13.25$, $D = -0.001$

(c) Underdamped response. $P = 0.04$, $I = 5.46$, $D = 0$

Figure 3.7: The different transient responses of the negative feedback control system for the SPPE.

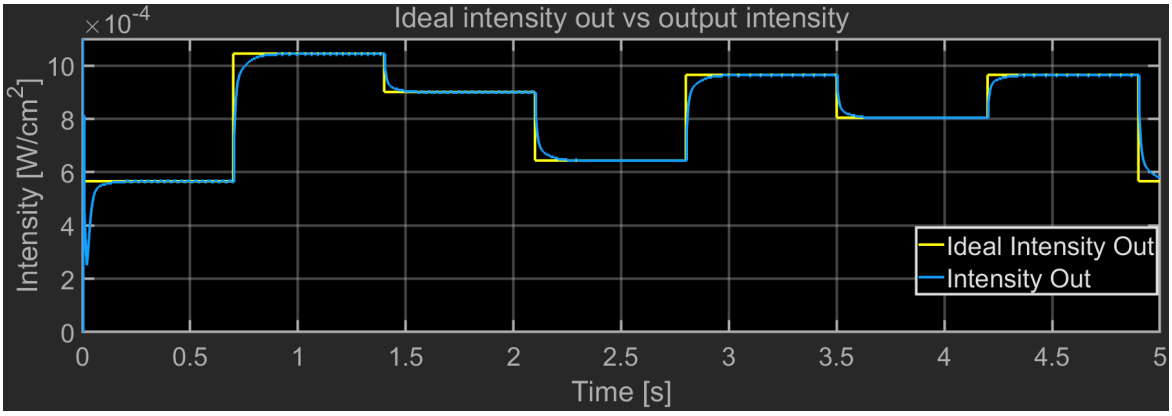


Figure 3.8: Critically damped input-output behaviour of the finalized system.

There are several systematic approaches in setting the PID parameters, i.e. to select the values of K_p, K_i, and K_d that lead to a stable control system behaviour ([27], Ch. 5, 6, and 7). The method used here is the Simulink Autotuner tool [29]. This tool makes a linear approximation of the *plant*, i.e. the blocks of the control loop besides the controller. The tool then allows the designer to specify the system transient response.

Several transient responses of the system have been made with different PID parameters. These responses are shown in Figure 3.7. The goal is to find a set of parameter values that satisfy the requirements. Such a set of parameters is given by the critically damped response of Figure 3.7b. There exists no overshoot in this response. The rise time is 0.06 seconds ($<0.10 \cdot t_0$) and a 0% steady state error occurs. The parameters of this response satisfy the requirements.

Hence, the following control parameters are selected to be used in the controller:

- $K_p = 0.23$
- $K_i = 13.25$
- K_d = -0.001

These parameters result in a variable step input-output behaviour as seen in Figure 3.8. The spike at the beginning of the output is due to an algorithmic artefact: the simulation expects initial conditions of the UVC LED system. Hence, the spike will not occur in a physical SPPE.

3.4 Control implementation

Implementation of the control loop can be performed in several ways. The most convenient way is a digital implementation as it allows for high accuracy, speed, and flexibility in the design, and it is low-cost ([27], Ch. 8), ([30], Ch. 1). Several layers of abstraction are possible for these digital control systems.

The lowest abstraction layer is the FPGA implementation. Such a system allows for low latency, high frequency, and parallel performing control systems. Its downside is a more complicated design compared with higher layers of abstraction.
The next abstraction layer uses a microcontroller to implement the control system. These microcontrollers are self-contained systems that include one or more processors, memory, and some peripherals such as an analogue-to-digital converter (ADC) and digital communication interfaces.
Microcontrollers can be programmed with several programming languages, starting with low-level assembly language up to the higher-level languages like C++. The choice for a programming language will mainly be based on the capabilities of the language and the experience of the programmer in specific languages.

3.4.1 Microcontroller choice

There are many commercial microcontrollers available. The following requirements are set up to select an adequate one for the Sensing and Control subsystem:

- The microcontroller has a small form factor.
- The microcontroller is low power or can use a sleep mode.
- The microcontroller has connections for the digital communication protocol I^2C.
- The microcontroller has non-volatile memory storage both for the program and for storing variables.
- The microcontroller has excellent documentation, example code and libraries as it is not possible to test the microcontroller during the project.
- The microcontroller has a low price, if possible.

Based on these requirements, the ATmega328P microcontroller by Atmel (see Figure 3.9) is selected for the Sensing and Control subsystem. Some data about the microcontroller is summarized in Table 3.2.

The ATmega328P is used in devices by Arduino. This company is known for being open-source and providing excellent documentation and example code [31]. Arduino devices are programmed using the software development tool Arduino IDE, which is programmable in C++ [32].

The ATmega328P contains a connection for I^2C communication. I^2C (Inter-Integrated Circuit in full) is a synchronous serial communication protocol that allows for multi-master and multi-slave setups. The protocol sends data bundled in packets. Its design uses two wires to which all masters and slaves can be connected. It allows for simple add-ons by giving each device a 7-bit address [33].
There are many libraries available that simplify the communication between the ATmega328P and sensors using I^2C. Therefore, sensors that communicate by I^2C are preferred.

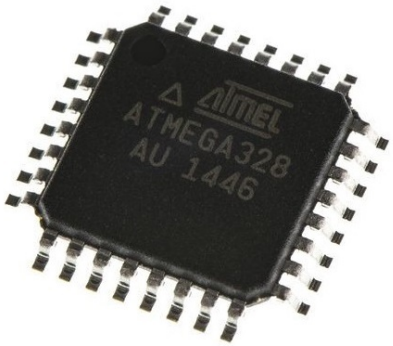


Figure 3.9: The Atmel ATmega328P microcontroller, its dimensions are (l, w, h): 7, 7, 1 mm [34].

Table 3.2: Specifications of the Atmel ATmega328P microcontroller [34].

Microcontroller characteristic	**ATmega328P**
Operating voltage	[2.7, 5.5] V
Clock frequency	8 MHz
Dimensions (l, w, h)	7, 7, 1 mm
Current usage	25 mA (active mode)
	0.57 mA (sleep mode)
Digital pins	14
Analog input pins	6 (using 10-bit DAC)
Memory	32 KB Flash
	2 KB SRAM
	1 KB EEPROM
Communication techniques	I^2C, SPI, USART

3.4.2 Microcontroller PID implementation

The easiest way to implement PID control in a microcontroller is by discretizing Equation (3.5). The discretized PID control equation is written as:

$$u[n] = K_p e[n] + K_i \sum_{i=0}^{n} e[i]\Delta t + K_d \frac{e[n] - e[n-1]}{\Delta t} \tag{3.6}$$

where Δt is the time between two consecutive controller computations. However, this equation is a too simplistic way of implementing the PID control [35].

In the SPPE, the PID control is implemented by using the Arduino PID Library written by Brett Beauregard (validated by Arduino) [35]. This library improves Equation (3.6) by solving the issues of derivative kick, reset windup mitigation, and initialization jumps. More information on these issues is provided by Beauregard [35].
The library allows for a straightforward easy initialization of the controller. It also allows us to set a range on the output value of the controller.

The PID control is implemented in the microcontroller by using the function as shown below. Here, `Imin` is the reference value that is an input of the control loop. The output is given as a PWM signal by using `analogWrite()` at the UVC-LED array that is specified by `selectUV`. The function is executed for `t0` seconds. The time that lines 6-8 take is taken into account with the aid of `PIDtimer` and `Tremain`.

```
1  void PIDcontrol(int selectUV, float Imin){
2      PIDref = Imin;
3      Tremain = t0;
4      while(Tremain > t0/100){
5        PIDtimer = millis();            // The current time
6        PIDin = getI(selectUV);         // Measure the UV intensity
7        myPID.Compute();                // Calculate PIDout
8        analogWrite(selectUV, PIDout); // Provide the output to the UVGI
9                                           subsystem
10
11       PIDtimer = millis()-PIDtimer;  // The time the PID control took
12       Tremain = Tremain - PIDtimer;  // The time remaining of t0
13     }
14   }
```

Now, the questions at the beginning of this chapter have been answered. The operation of UVGI was explained in Section 3.1. The UV intensity at the filter material was measured and included in a negative feedback control loop. From the reference value and the feedback, the error signal was calculated. The radiation was controlled by means of PID control. Lastly, the control was implemented in the ATmega328P microcontroller using the algorithm as shown above.

4 | Reference Function

As discussed in the previous chapter, the SPPE measures the UV intensity at the filter material. It compares this with a reference value. From the resulting error signal the controller calculates the target intensity and provides this to the UVC LED driver circuit. This signal indicates to either:

- Increase the intensity because the inactivation of pathogens is not strong enough. It is essential to give this indication in order to guard the safety of the user, as a too low inactivation can lead to infection or a false sense of safety.
- Or decrease the intensity because the inactivation is stronger than required. If the intensity can be lowered, the power consumption is lowered, and thus the battery life is extended.

The reference intensity I_{ref} is often presented as a single value. It equals the minimum dose to inactivate a certain factor of the virus population, divided by the time this dose is applied: $I_{\text{ref}} = I_{\text{min}} = D_{\text{min}} \cdot t_0^{-1}$. This equation is a simple derivation from Equation (3.1). The *min* subscript indicates that this is the minimum intensity or dose that needs to be applied in order to deem the filter to be *sterilized*.
However, according to Kowalsky ([3], Ch. 4), the I_{min} depends on the *ambient conditions*. These conditions concern the temperature (T) and the relative humidity (RH) of the air in the SPPE. It might be necessary to determine a number of values for $I_{\text{min}}(T, RH)$ at different ambient conditions.

How behaves such a function qualitatively and quantitatively? This is addressed in the following two sections. Next, a sensing system is designed in Section 4.3 to measure the ambient conditions. Last, the function is implemented in the microcontroller.

4.1 Qualitative reference function

Little research has been published regarding the dependence of I_{min} on ambient conditions. However, research has been conducted to study the effect of ambient conditions on the *pathogen viability* or on the *UV efficiency* in UVGI [36]-[46]. These can be linked to I_{min} in the following way:

> As the viability of pathogens increases, more UV light is needed to have the same level of inactivation, hence I_{min} increases. On the contrary, if the UV efficiency increases, less UV light is needed to have the same level of inactivation, hence I_{min} decreases.

It is thus possible to extract qualitative information on $I_{\text{min}}(T, RH)$ from scientific articles that do not consider $I_{\text{min}}(T, RH)$ specifically, but that consider virus viability or UV efficiency under changing ambient conditions ([3], Ch. 3).

The dependency of I_{min} on the ambient conditions differs between viruses and bacteria ([3], Ch. 3). Because of the current pandemic, the focus in this thesis will be on viruses, especially viruses of the corona type. However, to illustrate the potential broader application of the UVGI disinfection, the effect of ambient conditions on the UVGI of bacteria is discussed in the next subsection as well. The ambient conditions regarding bacteria are not considered in further depth after that. Please note, that the control loop is software driven, and it is possible to reprogram the SPPE's control loop to target specific pathogens. Nevertheless, it is good to keep in mind that the SPPE's design also enables the inactivation of other pathogens, such as bacteria.

Please note two aspects regarding the ambient conditions. The air pressure is not taken into account for the ambient conditions, because the pressure is not of importance for the viability of pathogens [36]. Furthermore, the RH is considered in order to determine the effect of the humidity on the viability of pathogens, because the RH is a better indicator than the absolute humidity for this purpose [37].

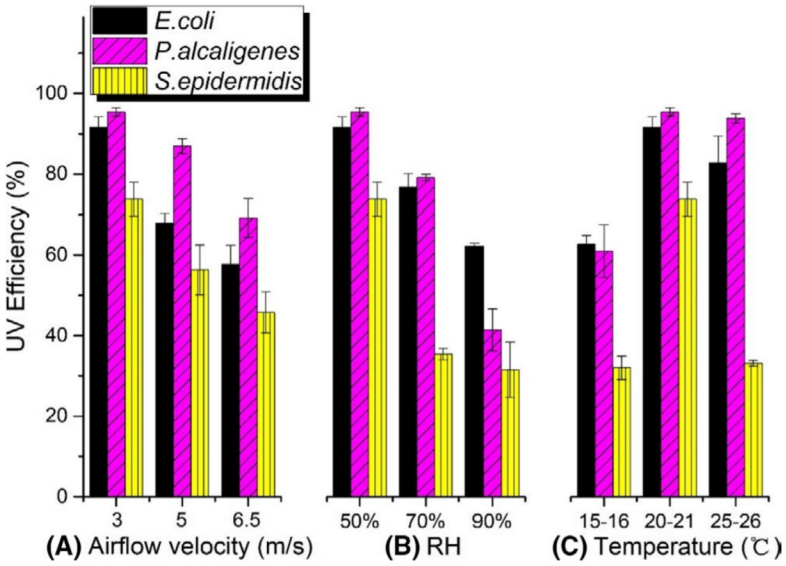


Figure 4.1: UV efficiency on three kinds of bacteria under changing ambient conditions [39].

4.1.1 Bacteria

Bacteria show a clear dependence on RH: for increasing RH, the viability increases and the UV efficiency decreases [38], [39]. The decrease in UV efficiency depends on the considered family of bacteria: the decrease can be both linear, or experience a sudden drop at low RH or high RH, as can be seen in Figure 4.1B

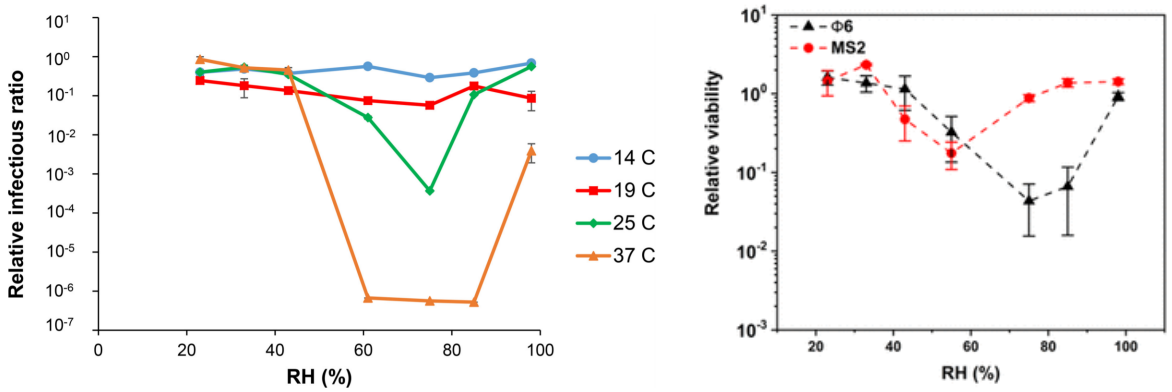
The dependence of UV efficiency on T is somewhat more complicated. For increasing T, the UV efficiency first increases, and then decreases. It depends again on the kind of bacteria, whether the increase and decrease occur at relatively low T or high T. There are also bacteria for which the dependence on T can be neglected [39], [40].

See Figure 4.1 for an indication of the UV efficiency for bacteria under different ambient conditions. This figure also shows a clear dependence of the UV efficiency on the airflow velocity. Obviously, the airflow in the SPPE depends on the physical design of the filter module. However, the unmasked airflow velocity for human breathing is not high enough to significantly affect the UV efficiency. The airflow velocity for human breathing in rest is in the range of 0.02-0.20 m/s [41]. The airflow velocity for deep breathing does not exceed 1.3 m/s [42]. The airflow velocity through the filter material is expected to be lower than the unmasked airflow velocity, as the filter material opposes some resistance to the flow [43]. The maximum airflow velocities are thus deemed low enough to neglect the effect of the airflow velocity on the UV efficiency.

4.1.2 Viruses

In general, the effect of T is inverse proportional to the virus viability, i.e. a higher T leads to a lower viability. The variation in virus viability depends on the virus species. An example of the viability of a virus is shown in Figure 4.2a. This figure concerns the ϕ6 virus, a commonly-used model for enveloped viruses. A virus is either enveloped or non-enveloped. Enveloped viruses, such as coronaviruses, protect their genetic material with an outer layer. Coronaviruses are also enveloped viruses. In the figure, the *virus infectivity* is used to measure its viability. This reproductivity measure is another measure for the viability of viruses.
Apparently, the effect of T is simple: a higher T means a lower viability, at least at an intermediate RH. However, the dependence on T becomes more complicated at low and high RH. Here, the viability at $T = 25$ °C is higher than the viability at $T = 19$ °C. Clearly, there is some interaction between T and RH. This interaction is also observed for the TGEV (transmissible gastroenteritis virus) and the MHV (murine coronavirus), which are both surrogates for the SARS-CoV [44].

The dependence of the virus viability on RH differs from that of bacteria. Most viruses experience a drop in their viability at an intermediate RH [36], [45], [44]. In other words, viruses survive best at low (<33%) and



(a) The effect of RH on the infectivity of the ϕ6 virus at different temperatures [37].

(b) The effect of RH on the viability of the ϕ6 and MS2 viruses [38].

Figure 4.2: Illustrations of the effect of T and RH on viruses.

at high (>95%) RH [38],[46], [47]. The intermediate RH at which the viability drop occurs depends on the virus species. The drop is a smooth, continuous process instead of some sharp decline. Figure 4.2b shows the viability of the ϕ6 virus and the MS2 virus, which is a commonly-used model for non-enveloped viruses.
The number of infections in specific countries also validates the dependence on RH during the SARS-CoV epidemic. Some countries have a more humid climate, i.e. a high average RH, which stopped the virus from spreading very fast [46].

4.2 Quantitative reference function

The book by Kowalsky ([3], Ch. 3) provides a simple mathematical model for the *fraction of survival S*. This fraction indicates how many pathogens survive a certain dose of UV light. The value of S is given by Equation (4.1):

$$S = e^{-kD} = e^{-kIt} \tag{4.1}$$

Here, k is the inactivation constant in [m^2/J], and use is made of Equation (3.1) for the dose D in [J/m^2]. When only one-millionth of the original pathogen population has survived ($S = 10^{-6}$), the treated surface is said to be *mathematically sterilized* ([3], p. 8). Often this value of sterilization is used, as it is difficult to fully sterilize a surface $\left(S = 0 \text{ when } \lim_{kD\to\infty} e^{-kD}\right)$.

In this model, it is actually k that depends on the ambient conditions, i.e. $k = k(T, RH)$ [48]. By Equation (4.1), the value of D needs to be adjusted to the value of $k(T, RH)$ in order to sustain mathematical sterilization. Hence, finding values or a function of $k(T, RH)$ results in a minimum value of I, the $I_{\min}(T, RH)$ as discussed before. The value of $I_{\min}(T, RH)$ is computed from Equation (4.2) and can be compared with the intensity measured by the SPPE's photodiode of Chapter 3.

$$I_{\min}(T, RH) = \frac{-\ln(S)}{t_0 k(T, RH)}\Big|_{S=10^{-6}} \tag{4.2}$$

4.2.1 k(T, RH) quantization

The article by Cutler *et al.* [48] is the only found source that gives an example of such a data set for $k(T, RH)$. The article provides the methodological techniques to construct a table for $k(T, RH)$, with different values for T and RH indicated in the rows and columns, respectively. Also, a 3x3 example of such a k-table is given. However, the article uses a linear dependence between D, I, and k, without specifying this any further. Hence, the article's table is not compatible with Equation (4.2). As all other sources discuss the exponential behaviour between D, I, and k, Equation (4.2) is deemed to be a better model than a linear model. Hence, we do not use the article's table in this thesis.

However, the article does illustrate the possibility of constructing k-tables for different kinds of viruses and

bacteria. The techniques and measurements needed to construct a k-table are beyond the scope of this thesis, but a k-table can be used for the reference function. Hence, in the remainder of this section, the use of an example k-table is illustrated. Here, the consideration is that a user of the SPPE can use a k-table to target some specific pathogen and then reprogram the SPPE to follow this k-table.

The k-table is considered for viruses of the corona type. From, ([3], App. B) it is known that:

$$k_{\text{coronavirus}}(T = 20°C, RH = 50\%) = 0.377 \ \frac{\text{m}^2}{\text{J}} \triangleq k_0$$

Where $k0$ is called the *classical inactivation constant*, i.e. the original inactivation constant that does not account for ambient conditions.

Based on the qualitative research of the previous section, an 11x11 k-table is created with entities in the range of $[0.7k_0, 1.3k_0]$. Measurement points are specified as $T = [0, 5, 10, ...45, 50]°C$ and $RH = [0, 10, 20, ...90, 100]\%$. Real-time measurements are rounded-off to the values specified in the table. The constructed k-table is shown in Table 4.1. Clearly, it would take a researcher way more time to measure 11^2 values for $k(T, RH)$ than to construct the 3x3 k-table of Cutler *et al.* [48]. However, for the illustrative purpose of the k-table in this thesis, it is deemed good to have a high resolution k-table with many entities.

Table 4.1: k-table ($\kappa(\boldsymbol{T}, \boldsymbol{RH})$) based on the qualitative research of Section 4.1. The value for k that will be used is calculated as k_0 times the factor indicated in the table, i.e. $k(T, RH) = k_0 \cdot k_{T,RH}$.

		Relative humidity [%]										
		0	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**100**
Temperature [°C]	**0**	0.70	0.70	0.75	0.75	0.80	0.80	0.80	0.80	0.75	0.75	0.70
	5	0.70	0.70	0.75	0.75	0.80	0.85	0.85	0.85	0.85	0.75	0.70
	10	0.75	0.75	0.80	0.80	0.85	0.90	0.95	0.95	0.95	0.85	0.80
	15	0.78	0.79	0.80	0.85	0.90	0.95	0.95	1.00	1.00	0.90	0.80
	20	0.85	0.86	0.88	0.90	0.95	1.00	1.05	1.13	1.15	1.13	0.95
	25	1.00	1.00	1.03	1.06	1.09	1.10	1.12	1.15	1.17	1.15	1.00
	30	1.08	1.10	1.12	1.13	1.14	1.15	1.17	1.18	1.18	1.17	1.08
	35	1.10	1.13	1.15	1.15	1.16	1.17	1.19	1.20	1.20	1.19	1.15
	40	1.13	1.15	1.17	1.17	1.20	1.23	1.24	1.25	1.26	1.23	1.18
	45	1.15	1.16	1.18	1.21	1.23	1.27	1.27	1.28	1.28	1.26	1.20
	50	1.16	1.17	1.20	1.23	1.25	1.28	1.29	1.30	1.30	1.30	1.22

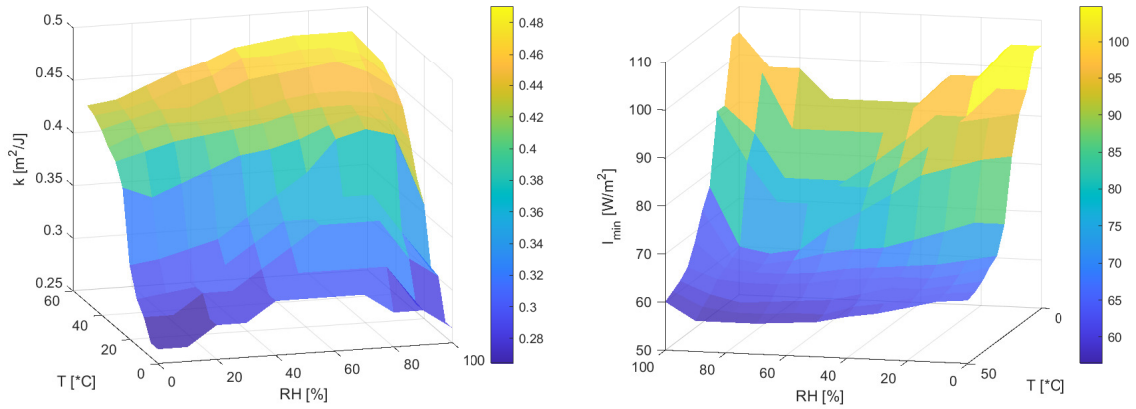
4.2.2 Reference function implementation

In order to implement the reference function in the SPPE, Equation (4.2) is first rewritten in matrix form:

$$\text{I}_{\text{min, 11x11}}(\mathbf{T}, \mathbf{RH}) = \frac{-\ln(S)}{t_0 \text{K}_{\text{11x11}}(\mathbf{T}, \mathbf{RH})}\Big|_{S=10^{-6}} \tag{4.3}$$

where bold letters indicate vectors and square letters indicate matrices. The size of the matrix is indicated in subscript.
A script is written in MATLAB to implement Equation (4.3) with the k-table of Table 4.1. This script is used to illustrate the functionality of the reference function by creating a plot of $\text{I}_{\text{min}}(\mathbf{T}, \mathbf{RH})$. The Matlab script is included in Appendix B.1. Figure 4.3a illustrates the difference in k-values in Table 4.1. Figure 4.3b shows the corresponding values of $\text{I}_{\text{min}}(\mathbf{T}, \mathbf{RH})$ on the given intervals for $\mathbf{T}$ and $\mathbf{RH}$.

The figures show the most important characteristics for the k-table: k increases for increasing T, and k shows an almost-parabolic behaviour for increasing RH. The effect of RH is stronger at lower T. Also, the figures show the inverse dependence of $\text{I}_{\text{min, 11x11}}(\mathbf{T}, \mathbf{RH})$ on $\text{K}_{\text{11x11}}(\mathbf{T}, \mathbf{RH})$, which is in accordance with Equation (4.3).



(a) The used values for $\kappa(\boldsymbol{T},\boldsymbol{RH})$

(b) The reference function $I_{min}(\boldsymbol{T},\boldsymbol{RH})$.

Figure 4.3: Surface plots of the reference function based on $S = 10^{-6}$ and $t_0 = 5$ seconds. $\kappa(\boldsymbol{T},\boldsymbol{RH})$ is given by Table 4.1. Note that the plots are rotated with respect to each other to provide the clearest figures.

4.3 Temperature and relative humidity sensor

The goal of the temperature and humidity sensor is to accurately relay the environmental information near the filter material back to the control system, i.e. back to the microcontroller. Both the sensor elements have to satisfy the requirements stated in Chapter 2. Where the mandatory requirement of reusability (M.V), the requirement of reference value calculation based on temperature and relative humidity (M.iv), and the design requirement of minimal power usage (D.IV) are leading.
Based on these primary requirements, the following list of criteria is proposed to which the sensor must adhere:

- The sensor must sense the T on the range of [0, 50] °C.
- The sensor must sense the RH on the range of [0, 100] %.
- The sensor has a small form factor to integrate it in the SPPE: the smaller it is, the better.
- The sensor has excellent documentation.
- The sensor is not sensitive to other conditions of the SPPE (UV, pressure, variations on).
- The sensor is as cheap as possible.
- The sensor preferably senses both T and RH.

Please note, regarding the temperature the assumption is made that the SPPE is used in indoor environments. As no physical tests can be performed, excellent sensor documentation is required. It is preferred to have one sensor to measure both T and RH, as this allows the measurements to be performed at the same location in the SPPE. Hereby the uncertainty of spatial variations is removed.

(a) The Si7021 temperature and relative humidity sensor, its dimensions are (l x w x h): 3 x 3 x 1.2 mm [49].

(b) The example PCB of the Si7021. Source: J.J.M. Lut, adapted from [49].

Figure 4.4: The Si7021 example PCB, and its schematic.

Table 4.2: Product information of the Si7021-A20 by Silicon Labs [49].

Sensor characteristic	**Si7021-A20**
RH range and accuracy	[0, 100] % ± 3%
T range and accuracy	[-10, +85] °C ±0.3°C
Output	I^2C
Response time	0.7s (T), 18s (RH)
Dimensions (l, b, h)	3, 3, 1.2 mm
Current consumption	0.06 μA (standby)
	150 μA (measurement)
	3500 μA (communication)
Protection cover	Teflon (PTFE)
Operating voltage	1.9 to 3.6V

4.3.1 Sensor choice

With the criteria known, a search for the correct sensor is performed. Many different sensors and manufactures are evaluated as a large marked is available for T and RH sensors. Some of the models evaluated are the HYT-939 (by Innovative Sensor Technology) [50], P14 Femtocap-G (by Innovative Sensor Technology) [51], ChipCap 2-SIP (by Telaire) [52], Si7021-A20 (by Silicon Labs) [49], and the SHT35-DIS (by Sensirion) [53]. See Appendix B.2 for an overview of the sensors and a comparison between them. From these sensors the Si7021-A20 and the SHT35-DIS is deemed the most promising and are compared more in-depth in Appendix B.3.

Based on the comparisons, the Si7021-A20 (Si7021 in short) temperature and humidity sensor is selected. This sensor is not expensive and draws little current. It has excellent documentation, is protected against UV light and water, and is applicable in the specified ranges for T and RH. See Figure 4.4a for an image of the Si7021 and Table 4.2 for a list of sensor characteristics regarding the Si7021.

Please note that two sensors are used per filter module to provide optimal UVGI control on both sides of the filter material (Requirement (M.ix)).

4.3.2 Sensing circuit

The implementation of the sensor in circuit form is quite straightforward because of the excellent documentation provided by the manufacturer [49]. One downside is the fixed I^2C address of the Si7021. As two Si7021 sensors (Requirement (M.ix)) with equal, fixed addresses are used per filter module, a communication conflict will occur on the I^2C bus.
This problem can be solved in software using a technique called *bit banging*. However, a more robust technique in hardware is considered. This technique uses an I^2C multiplexer (MUX), which allows communication with devices of identical I^2C addresses on the same I^2C bus. The TCA9548A of Texas Instruments is commonly used, as it is a cheap and dedicated I^2C MUX [54]. Therefore, the TCA9548A is selected to be used in the SPPE.

The I^2C bus requires connections to the voltage supply by means of pull-up resistors. The Si7021 documentation provides two equations to calculate the range in which the values of the I^2C bus pull-up resistors must fall, see Equations (4.4) and (4.5).
In Equation (4.4), V_{cc} = 3.3 V is the voltage supplied to the system, $V_{OL,max}$ = 0.6 V is the low output voltage, and I_{IO} = 3mA is the low output current. In Equation (4.5), τ_{rise} = 300 ns is the sensor rise time, and C_b = 26 pF is the bus capacitive load. These values are supplied by the sensor documentation or are estimated with the use of general I^2C documentation [55].
The resulting range of the pull-up resistors is [0.900, 13.6] kΩ. The value is selected to be 7.5 kΩ, approximately in the middle of this range and it is part of the E24 standard resistor series [56].
The circuit diagram of Figure 4.5 is used to connect one Si7021 sensor to the other electronics.

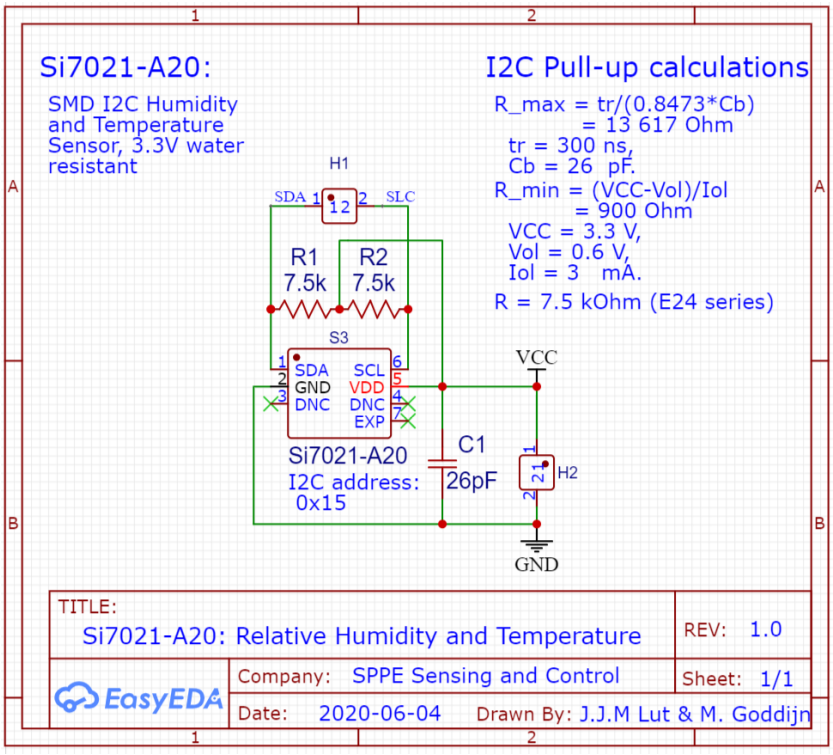


Figure 4.5: The Si7021 sensor schematic.

$$R_{\text{p, min}} = \frac{V_{\text{cc}} - V_{\text{OL,max}}}{I_{\text{OL}}} = \frac{3.3 - 0.6}{3 \cdot 10^{-3}} = 900\ \Omega \tag{4.4}$$

$$R_{\text{p, max}} = \frac{\tau_{\text{rise}}}{0.8437 \cdot C_b} = \frac{300 \cdot 10^{-9}}{0.8473 \cdot\ 26 \cdot 10^{-12}} = 13.6\ \text{k}\Omega \tag{4.5}$$

4.3.3 Sensor placement and implementation

The air near the filer material is quite a harsh environment for sensitive sensors, due to the UV intensity and the high humidity caused by breathing. A small study is performed to maximally protect the sensor. Two documents provide insight in the optimal sensor placement: the designer's guide of the Si7021 [57], and the user guide of the previously discussed SHT35-DIS, which is a very similar sensor that also uses the same protective cover as the Si7021 [58].

Main considerations to be taken for optimal temperature measurement:

- Avoid direct irradiation from a heat source (the Sun, for example).
- Protect against heating from other components or implement a compensation for thermal effect.
- Insulate the system's ground connection from large thermal masses in the system.

Main considerations for optimal relative humidity measurements:

- Protect the sensor against liquids using the protective cover.
- Avoid excessive porous materials near the sensor as they can absorb moisture from the air, and thus create errors in the measurements.

Please note that the filter material is porous, which is unavoidable.
To protect the sensor against heat sources, a long and slim printed circuit board (PCB) is developed. This PCB is designed to thermally isolate the sensor from the rest of the electrical system. The PCB is shown in Figure 4.4b.

4.3.4 Measurement algorithm

The reference function is implemented in the Arduino IDE as shown below. Please note that the function gets an argument, namely which Si7021 sensor is selected (on the inside or on the outside of the filter material). First, the measurements of RH and T are performed. The value of RH is determined first, as this measurement takes the longest to perform (18 seconds for RH and 0.7 seconds for T). The values are rounded off to match the k-table. The subscript `dis` is an abbreviation for *discretized*.
To clarify, the response time of the sensor is 18 seconds for RH and 0.7 seconds for T. Hence, when the sensor value is readout during these periods, the sensor value will be an average of the quicker changing conditions. Then, using the values of discretized values of RH and T, the associated value of k is selected from the k-table. The minimum intensity is calculated using Equation (4.2).

The program will temporarily end in case a measurement is taken that is not within the range of the k-table, in the worst case $I_{\text{min}} = 0\ \frac{\text{mW}}{\text{cm}^2}$. The health of the user may never be endangered (Requirement (M.VIII)), hence the following solution is developed:

> If the value of either T or RH is not within the k-Table, the minimum value of k is assigned. From Equation (4.2), the minimum value of k will result in the maximum reference value of I_{max}. This maximum value ensures that the filter material will receive the maximum dose, as a result of this ensuring the safety of the user.

```
1  float referenceFunction(int selectSi7021){
2    RH = getRH(selectSi7021);                  // Measure RH
3    T = getT(selectSi7021);                    // Measure T
4    RHdis = round(RH/RHinterval);              // Round off to the intervals
5    Tdis = round(T/Tinterval);                 // Round off to the intervals
6    if(0<=Tdis<kSize && 0<=RHdis<kSize)
7    {kVal = kTable[Tdis][Tdis];}               // Determine which value for k
       is used
8    else{kVal = 0.7*k0;}
9    Dmin = -1*log(S0)/kVal;                    // [J/m^2] Minimum dose
10   Imin = Dmin/t0;                            // [W/m^2] Minimum intensity
11   return(Imin);
12 }
```

4.3.5 Definitive control Loop

After implementation of the reference function, the control loop of Chapter 3 is updated. This results in the definitive control loop of Figure 4.6. Note that its behaviour is the same, only the input is now a variable step input, which is similar to Figure 3.8.

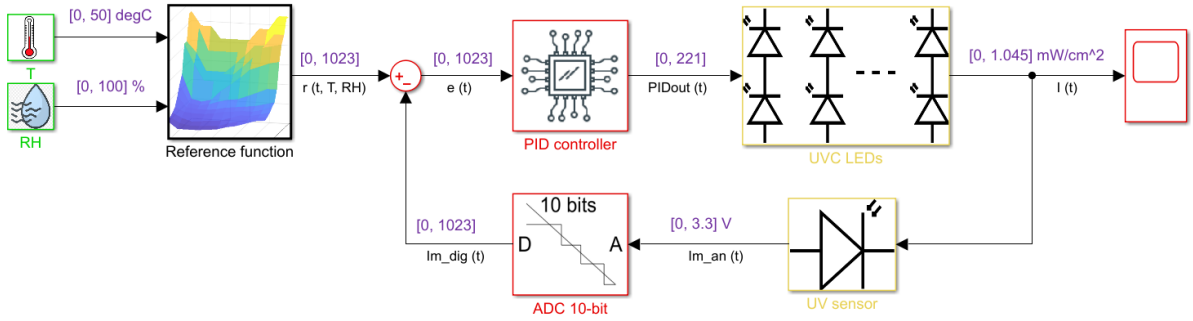


Figure 4.6: The definitive control loop as it is implemented in Simulink with the reference function.

5 | Filter Quality

In the current situation, with single-use face masks, medical workers dispose of face masks after a work shift of two hours. Each medical worker thus uses at least four face masks on a regular working day of eight hours. This produces contaminated waste, and a risk of infection when the mask and waste are not handled properly. Moreover, many countries experienced a shortage of face masks at the beginning of the COVID-19 pandemic [1].
Here, the SPPE offers a solution with its reusable filter. This filter material is kept inside the SPPE throughout one or multiple working shifts. However, micrometre-size particles will get caught in the filter material as the SPPE is used over time. Eventually, this will result in a reduced filter capacity, which is less safe to use. Hence, a smart filter quality function should indicate when the filter material of the SPPE needs to be replaced.

This consideration raises the following questions: when is the filter capacity reduced? How is this measured? The answers to these questions are discussed in Section 5.1 and 5.2. The implementation of the filter quality function is also discussed.

5.1 Filter quality

As the filter material gets clogged, the user needs to put more effort into breathing. Physically, the pressure drop over the filter increases [59]. So if the pressure drop is measured in the SPPE, the clogged filter can be detected. The filter is deemed to be clogged if the pressure drop is higher than a specified threshold.

5.1.1 Pressure threshold

There exist several medical guidelines for the maximum pressure drop. The new guideline of the European Union [60] prescribes different pressure thresholds for different amounts of air that is flowing through the filter material. This amount of air is referred to as the *air flow* through the filter material. Table 5.1 shows the pressure thresholds of the European Unions guidelines for different airflows.
Therefore, to determine the correct threshold, the SPPE must also be able to determine the airflow through the filter. In the next subsection, the procedure is discussed to determine the airflow.

Table 5.1: European Union guidelines for the pressure threshold of face masks.

Pressure threshold [Pa]	Air flow [L/min]
70	35
325	95

5.1.2 Air flow calculation

The general equation for flow is:

$$Q = Av \tag{5.1}$$

Where Q is the flow in $\frac{\mathrm{m}^3}{s}$, A is the flow area in m^2, and v is the velocity of the flow in $\frac{\mathrm{m}}{\mathrm{s}}$. An expression for the flow velocity is derived from the general equation of dynamic pressure:

$$P_{\mathrm{dyn}} = \frac{1}{2}\rho v^2 \tag{5.2}$$

Where P_{dyn} is the dynamic pressure in Pa and ρ is the density of the flow in $\frac{\mathrm{kg}}{\mathrm{m}^3}$.

In the case of the SPPE, the dynamic pressure is represented by the pressure drop over the filter material, ΔP. The density of the flow is equal to the density of air in the filter, ρ_{air}.

While the air flows through the channels of the SPPE it will encounter the filter material contained inside the module. This filter material will oppose the flow of air and act as a resistance to the flow. Incorporating this resistance in the airflow calculation is done by introducing a coefficient to account for this. This coefficient is called the discharge coefficient, C_d. It is a dimensionless coefficient in the range of [0, 1].

By combining Equation (5.2), Equation (5.1), and the discharge coefficient C_d results in Equation (5.3). This is the orifice equation from fluid mechanics ([43], p. 1153). The flow of air through the SPPE is approximated using this equation.
The parameters of the equation are determined below.

$$Q = AC_d\sqrt{\frac{2\Delta P}{\rho_{\text{air}}}} \tag{5.3}$$

Orifice equation quantification

The UVGI team determined that the minimal filter diameter 8 cm, such that $A = 50\ \text{cm}^2$.
The discharge coefficient C_d needs to be determined for the specific filter material that is used within the SPPE. However, the value of C_d could not be found for any filter material. It is known that $C_d \in [0,1]$, and usually $C_d < 0.5$ for small-sized orifices ([43], p. 1153), ([61], no page). The article by Wang *et al.* [62] provides a value for the discharge coefficient of monofilament-woven fabrics, namely $C_{\text{d,Wang}} = 0.2$. As filter materials consist of multiple layers of fabric, we expect that $C_d < 0.2$ for the SPPE. In the remainder of this thesis, it is assumed that $C_d = 0.02$. However, we emphasize that this value should physically be determined for the filter material.
The pressure drop ΔP is not known beforehand, and will need to be measured.
The density of air ρ_{air} has the nominal value of $1.225\ \text{kg/m}^3$. However, in general ρ_{air} is dependend on several parameters. $\rho_{\text{air}} = \rho_{\text{air}}(T, RH, P_{\text{abs}})$, where P_{abs} is the absolute pressure. Using these parameters value of ρ_{air} is determined by the CIPM-2007 equation for the density of air [63]. This equation is non-linear and consists of many terms.

Finally, by inserting $A = 50\ \text{cm}^2$, $C_d = 0.02$, and $\rho_{\text{air}}(T, RH, P_{\text{abs}})$ in Equation (5.3) results in the equation for the airflow in the SPPE:

$$Q(\Delta P, T, RH, P_{\text{abs}}) = 8.485\sqrt{\frac{\Delta P}{\rho_{\text{air}}(T, RH, P_{\text{abs}})}}\ \left[\frac{\text{L}}{\text{min}}\right] \tag{5.4}$$

5.2 Sensing system

Clearly, several ambient variables have to be measured to determine the flow Q from Equation (5.4) and to determine the filter quality. These variables are: $\Delta P, T, RH,$ and P_{abs}. Fortunately, both T and RH are already being measured by the Si7021 sensor, as discussed in Chapter 4. These measurements can also be used for the filter quality purpose. The two remaining parameters, differential pressure ΔP and the absolute pressure P_{abs} are still unknown. In the following sections, a method will be presented to determine these values.

5.2.1 Sensor choice

Two measurement setups can be used to obtain ΔP and P_{abs}:

1. One absolute pressure sensor is used in combination with one differential pressure sensor, which measures ΔP directly. This is referred to as Setup 1.
2. Two absolute pressure sensors are used, one inside the SPPE and one outside the SPPE. The differential pressure is then the difference between these measurements $\Delta P = P_{\text{in}} - P_{\text{out}}$. Referred to as Setup 2.

In Figure 5.1 an overview of the two possible setups is given. In Setup 1 the differential pressure sensor is positioned such that it is able to measure the pressure at both sides of the filter material and the absolute pressure sensor is connected to the outside of the mask such that it can measure the atmospheric pressure. In Setup 2 the two absolute pressure sensors are positioned on both sides of the filter material.

In order to select the best setup for the SPPE, it is important to keep in mind that the smallest threshold

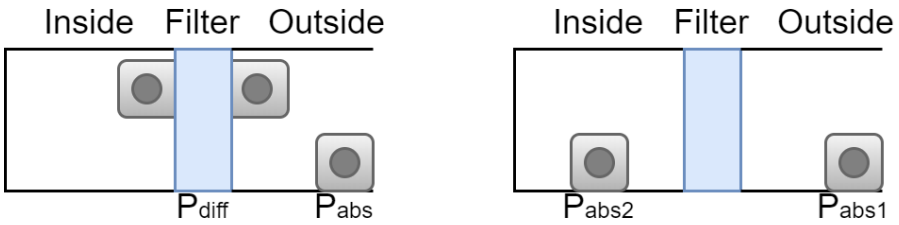


Figure 5.1: A diagram of both setups. Here, *inside* means the space between the filter material and the face, and *outside* means the space on the other side of the filter material. The left diagram is Setup 1 and the right diagram is Setup 2.

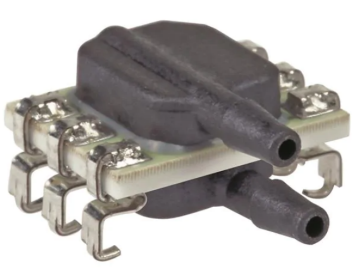
of Table 5.1 is only 70 Pa. The value of ΔP thus needs to be determined with high accuracy. The commercially available absolute pressure sensors with a small form factor have an accuracy of ±100 Pa, which is too large for this application. For this reason, Setup 1 is selected for the SPPE.

Besides the accuracy requirement for ΔP, the sensors also have some other requirements:

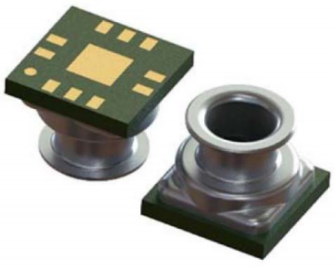
- The differential sensor is airtight.
- The sensors use I^2C to communicate with the microcontroller.
- The sensors heights are maximally 15 mm.
- The sensors are water resistant, so they can withstand the condensation caused by breathing.
- The sensors can withstand a large pressure spike. Such a spike may occur, for example, when the user sneezes or exhales sharply.

To clarify the first requirement: we found that there are two types of differential pressure sensors. The first type consists of a parallel plate capacitor which bends in the direction of a lower pressure. This type is airtight. The second type actually measures the airflow and calculates ΔP from this, such as the SDP3x (by Sensirion) [64]. Clearly, this type is not airtight. The SPPE's differential pressure sensor needs to be airtight, otherwise harmful pathogens may traverse through the sensor. This endangers the health of the user and is conflict with Requirement (M.III).

Taking the above requirements into consideration, the following sensors have been selected for the application. The ABPMRRV001PD2A3 (Honeywell) is used for the differential pressure sensor [65]. The LPS33HWTR (ST) is used for the absolute pressure sensor [66]. The sensors are shown in Figure 5.2, Table 5.2 lists relevant characteristics of these sensors. For an overview of pressure sensors that were considered, please see Appendix C.1 and Appendix C.2.



(a) The ABPMRRV001PD2A3 differential pressure sensor by Honeywell, its dimensions are (l x w x h): 12 x 11 x 6.2 mm [65].



(b) The LPS33HWTR absolute pressure sensor by ST, its dimensions are (l x w x h): 3.3 x 3.3 x 2.9 mm [66].

Figure 5.2: The SPPE's pressure sensors.

Table 5.2: Specifications of the ABPMRRV001PD2A3 and the LPS33HWTR [65], [66].

Sensor characteristic	**ABPMRRV001PD2A3**	**LPS33HWTR**
Sensor type	Differential pressure	Absolute pressure
Communication	I^2C, 0x28 address	I^2C, 0x5D address
Measurement time	0.46 ms	13 ms
Operating voltage	3.3 V	[1.7, 3.6] V
Dimensions (l, w, h)	12, 11, 6.2 mm	3.3, 3.3, 2.9 mm
Range	[0, 1] psi = [0, 6894.8] Pa	[260, 1260] hPa
Accuracy	± 17 Pa	±1.0 hPa
Current consumption	3.1 mA, 1 μA (sleep mode)	15 μA, 1 μA (sleep mode)

5.2.2 Sensing circuits

The ABPMRRV001PD2A3 absolute pressure sensor does not require additional electronics [65]. The sensor has an unused I^2C address, and can thus be connected to the microcontroller's I^2C bus without using the I^2C MUX from Chapter 3. Because of the relatively bulky design of the differential pressure sensor, it is placed on the main control board.

The LPS33HWTR differential pressure sensor requires a small number of additional components [66]. The circuit schematic is shown in Figure 5.3. The I^2C address of the sensor can be changed by connecting the CS pin to ground or V_{cc}. For this implementation, it is set to default address 0x5D.

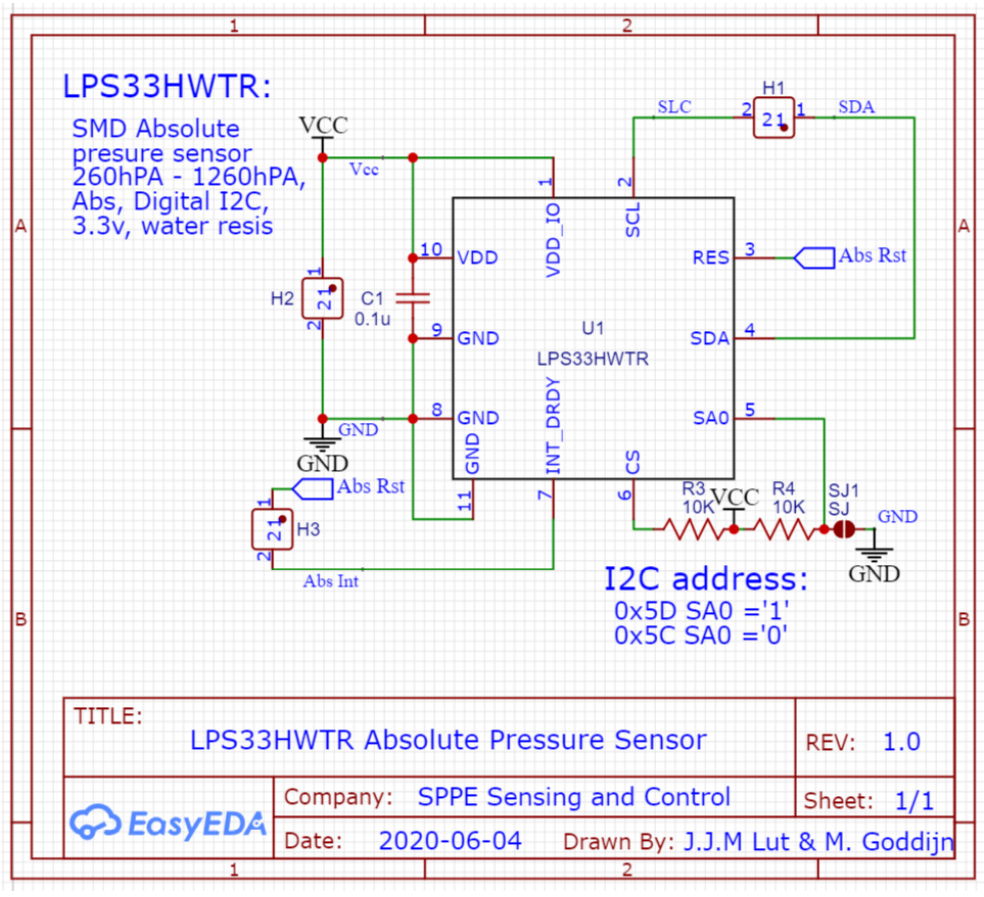


Figure 5.3: Schematic of the LPS33HWTR absolute pressure sensor.

5.2.3 Sensor integrations and implementations

To ensure optimal sensor functionality, they have to be placed in such that the environment will not inflict with their operation. To achieve this, the manufacturers specify the sensor limits and recommend their placement.

The main technical notes of the ABPMRRV001PD2A3 differential pressure sensor are [65]:

- The recommended connection method of pressure zones is tubing or O-rings.
- It is recommended to use silicone for the tubing.

As mentioned, the ABPMRRV001PD2A3 is placed on the main PCB next to the microcontroller. There is space reserved for the tube that goes to the outside of the SPPE. The other opening remains unconnected, as to measure the internal pressure.

For the LPS33HWTR absolute pressure sensor the following points are noted [66]:

- The sensor placement should be as close as possible to the environment under test.
- The sensor opening should be as large as possible.

The LPS33HWTR application note provides a guideline for the sensor integration, see the figure in Appendix C.1.
The LPS33HWTR is placed just below the surface of the SPPE. The hole that is created for it is sealed using an airtight material, such as a resin.

5.2.4 Measurement Algorithm

To finally determine if the filter is in need of replacement, pressure measurement data is processed. This is done by the microcontroller in the following manner.

First of all, the values of RH and T are measured, as these are used in the calculation of the air flow Q. The measurement of RH takes the longest (18 s, [49]), hence this measurement is performed first.
Second, the function `peakPdiffDetection()` is used to determine the maximum pressure drop that occurs in one minute. For this, the differential pressure sensor measures the pressure drop every 37 ms. When the measurement time (17 ms) is added to this, the controller can measure 1200 times per minute. This time is selected as it allows to measure the breathing pattern accurately during multiple breath cycles.
Third, P_{abs} is measured, such that all variables for Q are known and it can be determined using Equation (5.4).
Fourth, the function `determineIndSignal(Q, PdiffMax)` determines if the thresholds of Table 5.1 have been violated. If this is the case, the `indicationSignal` is set high. The `indicationSignal` sets the colour of the RGB-LED to blue, hereby notifying the user that the filter needs to be replaced.

However, if the user takes just one deep breath or sneezes during the measurements, the pressure drop measurements will show a spike, and it lets the microcontroller detect that the filter is clogged. In other words, it triggers a false-positive. In order to prevent the false-positive problem, the following method is implemented in the `determineIndSignal(Q, PdiffMax)` function:

> If the value of ΔP violates the threshold value of Table 5.1, all measurements are performed once more. This includes the measurements of RH, T, and P_{abs} and it includes the calculation of Q. If the threshold is violated again during the second measurements, the `indicationSignal` is set high and the user is notified about the clogged filter. Using this method, most of the false-positives of short-timed deep breathing or sneezing will be avoided.

To verify the workings and to graphically illustrate the algorithm a Simulink model has been developed Figure 5.4, resulting in Figure 5.5. Here, the filter quality algorithm is displayed with false-positive avoidance.

```
void FilterQuality(){
    RH = getRH(busSi7021_1);                                    // Update the
        value of RH, arbitrary sensor
    T = getT(busSi7021_1);                                      // Update the
        value of T, arbitrary sensor, but same as for RH
    PdiffMax = peakPdiffDetection();                            // Measure the
        maximum pressure difference
    Pabs = getPabs();                                           // Measure the
        absolute pressure
    Q = airFlow(T, RH, Pabs, PdiffMax);                         // Calculate the
         air flow in the filter
    indicationSignal = determineIndSignal(Q, PdiffMax);         // Determine if
        the threshold has been reached
    setIndicationSignal(indicationSignal);                      // Set the
        indication signal
}
```

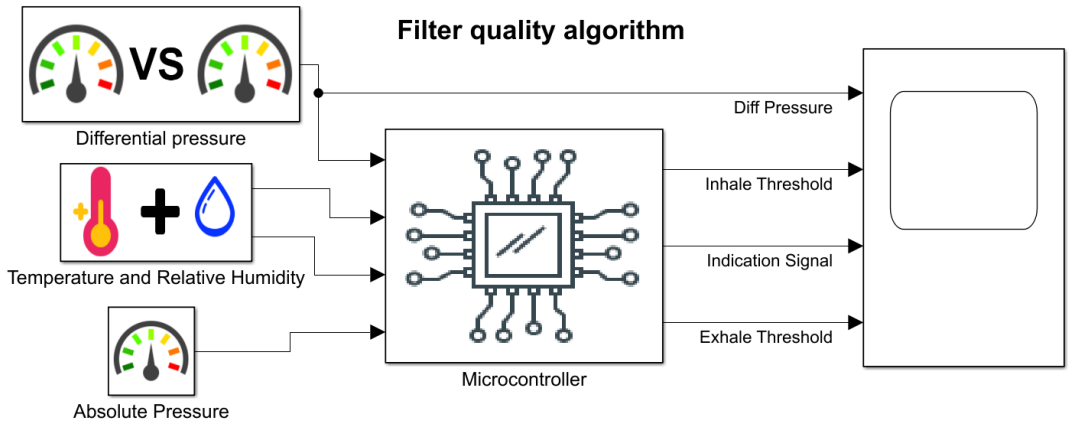


Figure 5.4: Simulink model of the filter quality algorithm. Inputs are the differential pressure, temperature and relative humidity and the absolute pressure.

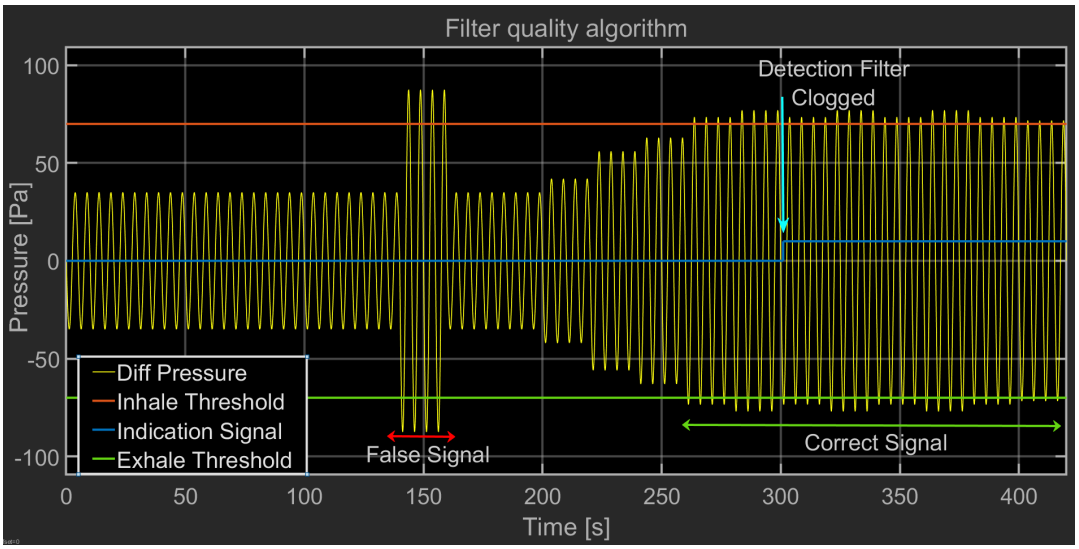


Figure 5.5: Filter quality algorithm with false positive avoidance. Yellow is the differential pressure of a person breathing. Orange and green are the threshold values calculated based on the different sensor inputs. Blue is the indication signal that triggers after two successive measurements.

6 | Sensing and Control Finalization

So far, techniques were developed to measure the UV intensity, temperature, relative humidity, absolute pressure and differential pressure. The UV intensity in the SPPE is controlled, and the filter quality is monitored by using these measurements. Now arises the question: how is a functioning prototype constructed from the developed techniques?

As mentioned in the Introduction, the current circumstances prohibit the teams from working on a physical prototype. The students aim for the most detailed design currently possible. In this way, a physical prototype can be realized with as little extra research and time as possible.
The most in-depth hardware design that can be delivered is a printed circuit board (PCB) design. The most in-depth software design that can be delivered is a C++ program written in the Arduino IDE. Both designs must not give any errors during the compilation. These designs are presented in the following two sections. After these sections, the design of the Sensing and Control submodule will be completed.

In Section 6.3, the total current consumption of the Sensing and Control submodule is calculated. In Section 6.4, a discussion is presented on how the system can be made more resistant against technical failures that occur after a prolonged period of use (several years). The resulting problems and possible solutions are presented. Lastly, several parameters need to be determined physically before the Sensing and Control submodule can function in the SPPE. These are discussed in Section 6.5.

6.1 Printed circuit board design

Two design considerations are taken into account for the design of the printed circuit board, namely:

- The printed circuit board must be as small as possible, with a maximum of 50x50 mm, as demanded by Requirement (M.ix).
- The SPPE must be designed circularly, as is demanded by Requirement (D.I).

The SPPE module consists of several layers each performing a different function, as seen in Figure 1.1. This layering requires the sensors to be standalone systems that can be positioned throughout the SPPE. For this, a small form factor is key. There is a separate PCB for the following: the UV sensor, the temperature and relative humidity sensor, the absolute pressure sensor, and the control system. The additional benefit of more boards is an increased level of circularity where failed sensors can be replaced individually.
The control system PCB will function as the main regulator board to which every other sensor board is connected. The largest components are located on this board, namely: the microcontroller, the I^2C MUX, and the differential pressure sensor.

The PCBs are designed based on their representative schematics, namely Figure 3.4, Figure 4.5, Figure 5.3, and Figure 6.1. The resulting control system PCB design is shown in Figures 6.2. The PCB designs of the three sensors are shown in Appendix D.1.

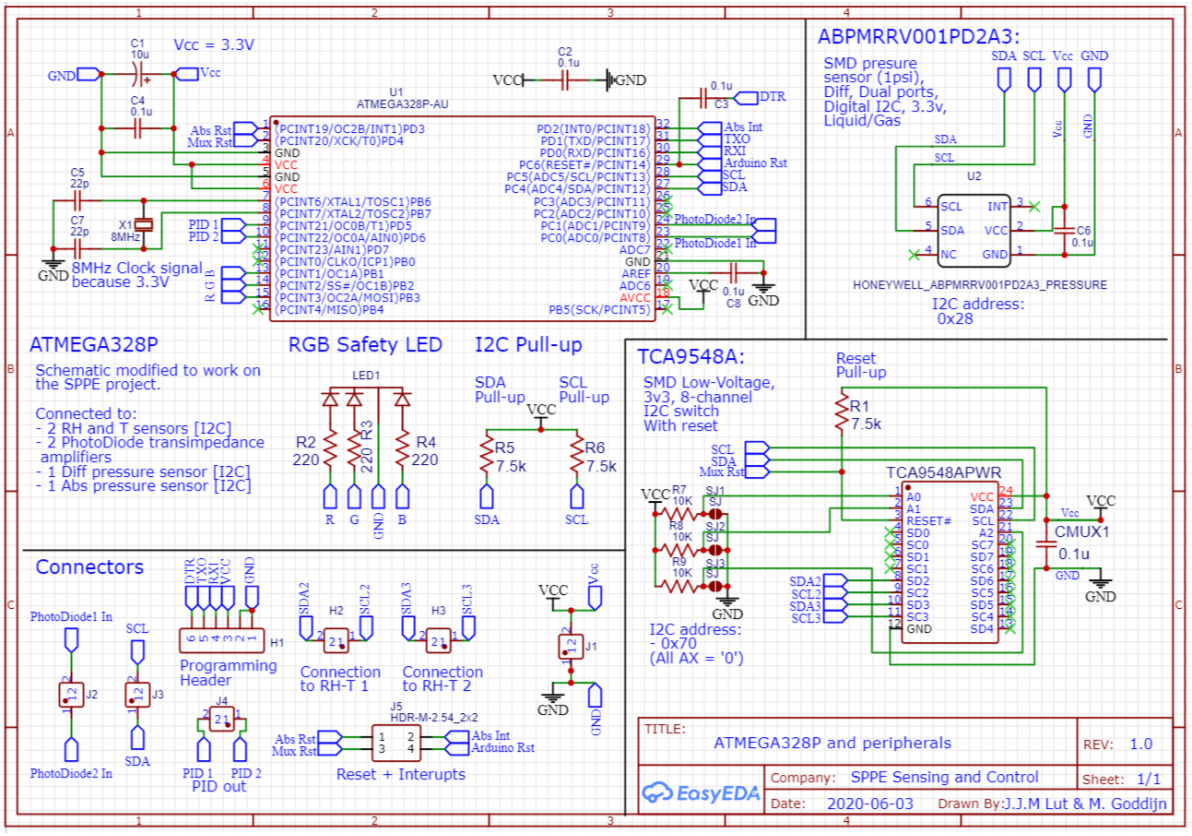
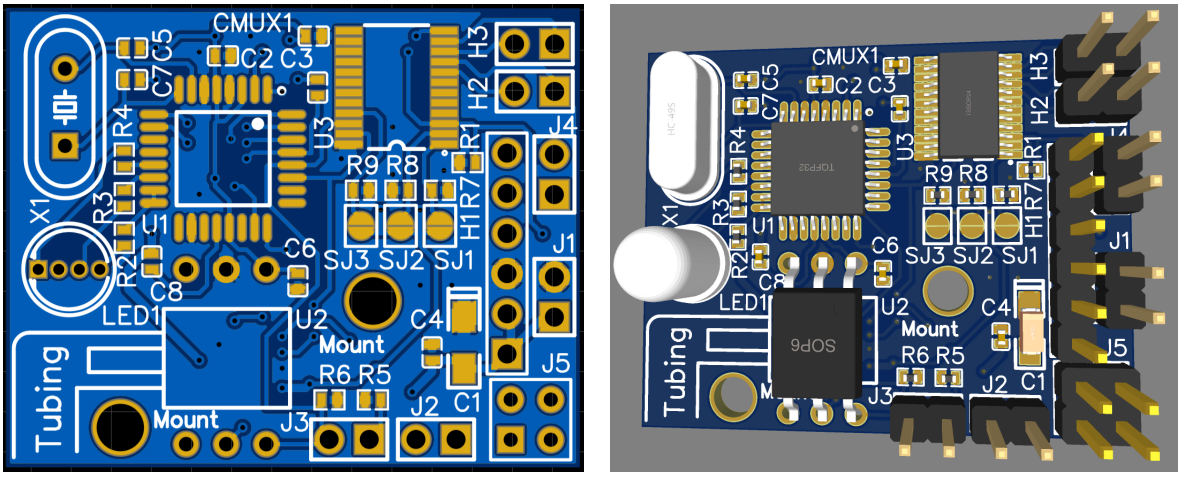


Figure 6.1: The circuit schematic of the control system. The upper-left corner shows the microcontroller. The upper-right corner shows the differential pressure sensor. The lower-left corner shows the connectors of the PCB. The lower-right corner shows the I^{2}C MUX to which the Si7021 sensors are connected.



(a) PCB design of the control system.

(b) Three-dimensional view of the control system.

Figure 6.2: In-depth design of the control system. The size of the PCB is 36x29x7 mm, including components.

6.2 Algorithm

The PID controller, the reference function, and the filter quality were already implemented in software in Subsection 3.4.2, Subsection 4.3.4, and Subsection 5.2.4, respectively. However, a more extensive computer program is needed to complete the Sensing and Control. The aspects of this program are discussed here. The complete program is included in Appendix D.2. The program compiles without errors or warnings. The program uses 20234 bytes (65%) of program storage space. The variables use 1139 bytes (55%) of the dynamic memory.

6.2.1 Sensor measurements

The functions to retrieve the measurements from the sensor are `getT()`, `getRH()`, `getPabs()`, and `getPdiff()`. These functions are already used in the functions discussed previously in this thesis. The function `TCA9548A()` is used to address the correct Si7021 sensor. In order to ease the I^2C communication with these sensors, several libraries were used that provide functions to communicate with the specific sensors. These libraries are:

- The Adafruit Si7021 Library 1.3.0, by Adafruit [67].
- The Adafruit LPS35HW Library 1.0.4, by Adafruit. This library can also be used for the LPS33HWTR that is used in the filter quality [68].
- Honeywell ABP Library 1.0.0, by ij96. This library can be used for the ABPMRRV001PD2A3 differential pressure sensor that is used in the filter quality [69].

6.2.2 Sleep mode

It is important to save power by enabling the sleep mode on the Atmega328P. The sleep mode decreases the current consumption from 25 mA to 0.57 mA, which is considerable. The sleep mode can be implemented in two methods [34]:

1. By enabling the so-called Power-down mode. In this way, the Atmega328P freezes the clock and thus disables all chip functionalities. The only way to "awake" the microcontroller is by applying an external interrupt.
2. By using the Watchdog Timer to go into a sleep mode for a short period, e.g. some seconds. The Watchdog "awakes" the microcontroller once the timer reaches the time-out value, which is specified before going to sleep. It is possible to go to sleep quickly again after awakening.

In practice, the first method allows the Atmega328P to sleep for a longer period of time. With this method, the current consumption can be lowered to a few μA [34]. However, an external interrupt needs to be provided. This signal can perhaps be provided by the microcontroller of the On-Board Power Management submodule. However, this is not implemented due to time constraints. To safeguard the user, it is also of the essence that the Atmega328P awakens. Hence, the decision is made to use the less ideal, but safe to implement, second method.
The sleep mode is implemented by using a while-loop:

```
1 while(sleepTime>10000){sleepTime = sleepTime-Watchdog.sleep(1000);}
```

The variable `sleepTime` is specified beforehand. The argument given to Watchdog.sleep() is the sleep time in milliseconds. The precise time the Watchdog Timer can run is in the order of some seconds [34]. To be safe, the time is set to one second. This has no further implications for the functionality of the sleep mode.

The sensors are also programmed such that they are in their respective sleep modes unless a measurement is requested.

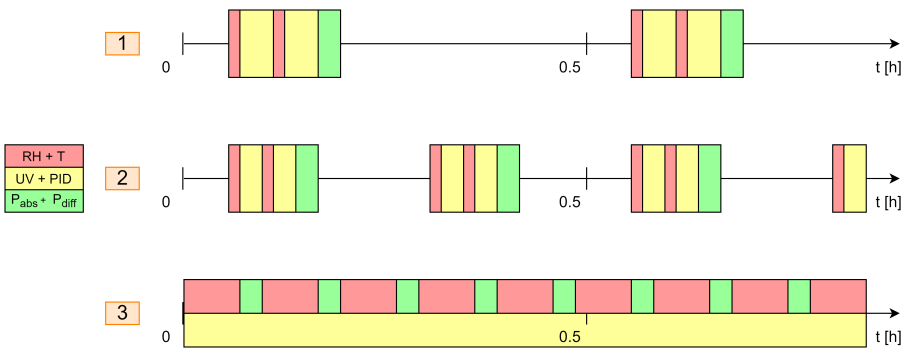


Figure 6.3: Various loop patterns for the Sensing and Control submodule.

6.2.3 Program loop

The functionalities of the Sensing and Control submodule are implemented in a loop in the program. The loop needs to do the following:

- Perform the `referenceFunction` for the first UVC LEDs,
- Perform the `PIDcontrol` for the first UVC LEDs,
- Perform the `referenceFunction` for the second UVC LEDs,
- Perform the `PIDcontrol` for the second UVC LEDs,
- Calculate the `airFlow` in the filter,
- Perform the `filterQuality` for one minute.

The loop can be implemented in various ways, as long as the radiation occurs once per half hour (Requirement (M.v)). Several *loop patterns* are shown in Figure 6.3. In Loop pattern 1, the UVC LEDs radiate once per half hour. In Loop pattern 2, the UVC LEDs radiate twice per half hour. Note that radiation times are shorter in loop pattern 2. To clarify the yellow block is drawn twice per irradiation moment this is because the LED arrays radiate sequentially. In Loop pattern 3, the radiation is continuous. The reference function is updated almost continuously. Many more patterns can be designed, the given patterns only illustrate the potential of the Sensing and Control submodule.

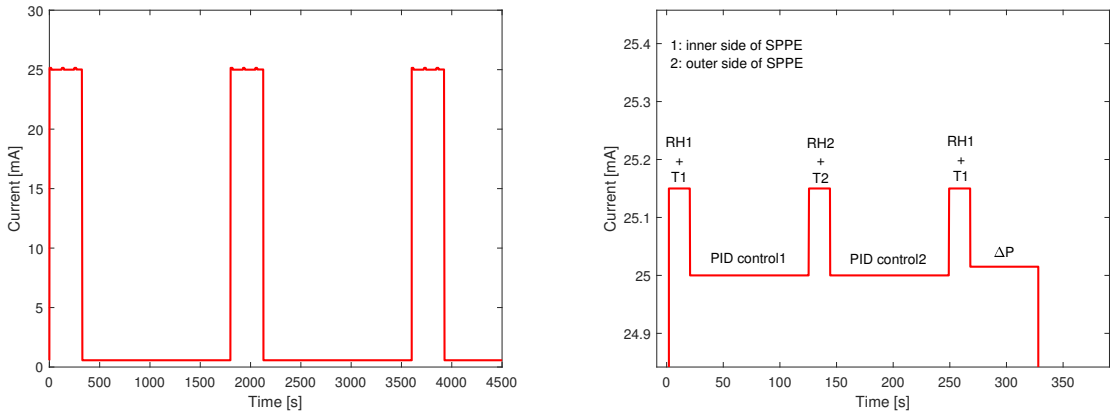
Loop pattern 1 is selected for the Sensing and Control submodule. The microcontroller is put in sleep mode for the longest time using this pattern, and hereby power is saved. As is by Design requirement (D. IV).

6.3 Total power consumption

The current consumption of the sensors were listed in Table 3.1, Table 4.2, and Table 5.2. The current consumption of the microcontroller is assumed to be 25 mA in active mode and to be 0.57 mA in sleep mode [70]. The total current consumption over time is shown in Figure 6.4. Note that current consumption of the pressure sensors is too small (for ΔP) or too short (for P_{abs}) to be visible in this figure.
The battery capacitance that is used during one cycle of the Sensing and Control submodule is given by Equation (6.1). Please note that, unlike in the rest of this thesis, I_c stands for current, and not for intensity.

$$\begin{aligned} t_{act} &= 2t_T + 2t_{RH} + t_{Pabs} + t_{\Delta P} + 2t_0 \\ &= 2\cdot 0.7 + 2\cdot 18 + 0.46\cdot 10^{-3} + 13\cdot 10^{-3} + 2\cdot 105 \\ &= 247.4 \text{ s} \end{aligned}$$

$$\begin{aligned} C_{cycle} &= (t_{cycle} - t_{act}) I_{c,ATmega,sleep} + 2t_T I_T + 2t_{RH} I_{c,RH} + t_{Pabs} I_{c,Pabs} + t_{\Delta P} I_{c,\Delta P} + t_{act} I_{c,ATmega,act} \\ &= (1800 - 247.4)\cdot 0.57\cdot 10^{-3} + 2\cdot 0.7\cdot 0.150\cdot 10^{-3} + 2\cdot 18\cdot 0.150\cdot 10^{-3} + 0.46\cdot 10^{-3}\cdot 3.1\cdot 10^{-3} \\ &\quad + 13\cdot 10^{-3}\cdot 0.015\cdot 10^{-3} + 247.4\cdot 25\cdot 10^{-3} \\ &= 7.08 \text{ As} = 1.97 \text{ mAh} \end{aligned} \tag{6.1}$$



(a) Current consumption during three cycles.

(b) Zoomed-in view on the active part of one cycle.

Figure 6.4: The Sensing and Control submodule's current consumption over time. The inner and outer sides of Figure 6.4b were defined in Section 5.2.

6.4 Technical failures

The Sensing and Control submodule is subject to component failures or other external technical issues that may occur. In Table 6.1 summarized several of these issues and their effect on the submodule. Below follows a list where the consequences and possible solutions to these problems are discussed. Please note, these functionalities are not implemented in the SPPE yet.

Table 6.1: Possible scenarios that disturb the functionalities of the Sensing and Control submodule.

What goes wrong?	**What will happen?**
1. Power cutoff	The whole SPPE will stop working, $I = 0 \frac{\mathrm{W}}{\mathrm{m}^2}$
2. Microcontroller breaks down	The submodule will stop working, $I = 0 \frac{\mathrm{W}}{\mathrm{m}^2}$
3. UV sensor breaks down	The UV intensity is not measured.
4. T+RH sensor breaks down	The temperature and relative humidity are not measured.
5. Absolute pressure sensor breaks down	The absolute pressure is not measured.
6. Differential pressure sensor breaks down	The differential pressure is not measured.

1. An RGB LED of the On-Board Power Management submodule will stop radiating on the SPPE, hereby notifying the user that the power is down. There are no further consequences for the Sensing and Control submodule: once the power is turned on again, the system restarts from the beginning without issues.

2. The UVC LEDs will not radiate. Hence the user must be notified. However, in this case, the user is not notified by the external LED, as the Power System submodule is still operational. The RGB LED of the Sensing and Control submodule will stop radiating, but this may not be seen by the user, as it only radiates during a short time.
   The problem can be solved by providing communication between the two subsystems, as a check to see if a response follows. If the On-Board Power Management submodule receives no response, the microcontroller is down, and the user is notified using the external LED.

3. If the intensity is not measured, or measured incorrectly, the error signal in the control loop will always indicate that the intensity is too low. This causes the PID controller to aim for maximum intensity, i.e. $I = 1.045 \frac{\mathrm{mW}}{\mathrm{cm}^2}$. Hence, the user is not endangered by a too low intensity and too low level of inactivation. The problem will cause the power consumption to increase.
   A possible solution to a broken UV sensor may be to judge if the sensor is broken based on the measurements. Numerous adjacent unexpected measurements indicate the sensor needs to be replaced. The replacement is installed relatively easily, as the sensor is based on a separate PCB. The user gets notified using the RGB LED.

4. If the T+RH sensor breaks down, the I^2C communication will also stop functioning. It is possible to detect this based on the absence of responses in the communication. The sensor is replaced relatively easily, as the sensor is based on a separate PCB. The user gets notified using the RGB LED.

5. Similarly, if the P_{abs} sensor breaks down, the I^2C communication will stop. By the absence of responses, the microcontroller will detect the failure of the sensor. The sensor is replaced relatively easily, as the sensor is based on a separate PCB. The user is notified using the RGB LED.

6. Similarly, if the P_{diff} sensor breaks down, the I^2C communication will stop. By the absence of responses, the microcontroller will detect the failure of the sensor. The user is notified using the RGB LED.

6.5 From idea to realization

At this point in the thesis, the theoretical design of the Sensing and Control submodule is completed. However, this design is not ready for actual product implementation. There are several items left to be designed and measured using a physical prototype. However, as mentioned in the Introduction, this is not possible under the circumstances of the current pandemic. In order to be as complete as possible, the unfinished design items are presented here. Once these items have been designed and measured, the Sensing and Control submodule is finished.

1. **System integration** In order to perform the following measurements, the SPPE prototype needs to be assembled in its entirety. For the Sensing and Control submodule, this includes: developing the PCBs, placing these in the SPPE, and uploading the algorithm onto the microcontroller.

2. **Gauging the UV sensor** It is important to check if the UV intensity at the sensor is indeed in the range of [0, 1.1] mW/cm^2. If it is far below this expected value, not the full voltage range is used, and resolution is lost. The design of the sensing circuit needs to be changed to obtain the best resolution. This is done by changing R and C to alter the transimpedance characteristics of the amplifier circuit.

3. **Measuring the *k*-table** The k-table of Chapter 4 is an estimation based on the literature review. The actual k-table needs to be determined by experts in the respective field. The new k-table also needs to be implemented in the algorithm.

4. **Measuring C_d** The discharge coefficient C_d of Equation (5.3) needs to be measured for the filter material that will be used in the SPPE. This can be done by connecting the prototype to an airflow meter, hereby measuring Q, for a given ΔP and ρ_{air}. Once C_d is calculated, its value is inserted in Equation (5.3).

5. **Testing the PID controller** To ensure that the control loop functions as intended, the input-output behaviour of the loop must be tested thoroughly. In the algorithm, code has been implemented to display values of the control loop on the serial monitor of the Arduino IDE. These values can be used to verify the functionality of the control loop.

6. **Testing the I^2C communication** The algorithm has been written without being able to test the communication with the sensors. Several libraries were used to make the I^2C communication easier to implement. The algorithm needs to be tested in the prototype to verify that the communication runs smoothly. Unfortunately, it was not possible to simulate the algorithm using online tools, as the I^2C communication is very sensor-specific, and the correct sensors are not available in online tools.

7. **Measuring the current consumption** The current consumption of Section 6.3 is an estimate, as it is difficult to estimate the actual current consumption of the Atmega328P. The current consumption of the Sensing and Control submodule needs to be measured in order to assure that the battery lifetime satisfies the requirements.

8. **Measuring the breathing pattern** The filter quality indication signal is triggered by measurements of the differential pressure that are above the threshold. These measurements are taken in two consecutive minutes. By measuring the breathing pattern through the SPPE, it can be assured that this is a correct way of setting the indication signal and monitoring the filter quality. If this way turns out to be unreliable, a different indication algorithm needs to be developed, by using peak detection, for example.

7 | Conclusion and Discussion

7.1 Conclusion

Project summary

To summarize what has been achieved in this project: first, in the Introduction the design of the SPPE has been presented and divided into three parts. These are the Ultra Violet Germicidal Irradiation, Sensing and Control, and On-Board Power Management.

In Chapters 3-5 the design of the Sensing and Control submodule has been developed. In Chapter 3, a discrete negative feedback control loop with PID control was designed. This control loop uses a photodiode transimpedance amplifier circuit as feedback and an ATmega328P as the microcontroller. The PID parameters were estimated with a Simulink model resulting in a critically damped system.
Chapter 4 was based on an insight from the literature study, namely the notion that temperature and humidity influence the amount of UV irradiation required. This transformed the reference value of the control loop into a reference function with parameters temperature and relative humidity as input. This function strives to find the optimum irradiation intensity, both saving power and ensuring the safety of the user.
In Chapter 5, a method of active filter monitoring was developed. An estimation is made on the airflow using the orifice equation from fluid mechanics. By measuring the temperature, relative humidity, differential filter pressure, and absolute atmospheric pressure, an estimation of airflow and filter quality can be made. This quality measurement is based on European guidelines.
In Chapter 6 the submodule's design has been finalized, and a list of unfinished design items was presented.

Project requirements

The Programme of Requirements has been split into two parts, the requirements applicable to the entire SPPE design and the requirements applicable to the Sensing and Control submodule.

Of the first part, the Sensing and Control submodule is subject to the Mandatory Requirements (M.III), (M.VII), (M.VIII), and (M.IX). The submodule has also been designed in accordance with the SPPE's design aspects in mind. The submodule satisfies Requirement (M.III), (M.VIII) and (M.IX). By the application of proper sensor sealing, a minimum required value for the UV intensity and useful feedback to the user. As of yet it remains unclear if Requirement (M.VII) is satisfied, as the expected lifetime is not mentioned on the datasheets of the used components. An attempt to contact the manufacturers is made, but their responses remain absent.

The second part of the requirements applied directly to the Sensing and Control submodule. The mandatory requirements (M.i)-(M.viii) are all satisfied by the very design of the submodule. The submodule has also been designed with the design aspects in mind. We like to emphasize Design Aspects (D.iii) and (D.iv), as this greatly enlarges the applications in which the submodule can be used. The algorithm is easily adjusted to make the submodule applicable to the sterilization of other pathogens. The whole design is easily applied in another UVGI product only requiring minor changes.

Problem definition

In the Introduction, the thesis' problem was defined in three main questions, repeated here for convenience:

1. How can we ensure the correct UV dose is applied to the filter material?
2. How can we use sensors near the filter material to adjust the UV intensity, hereby controlling the UVC LEDs in a power-efficient way?
3. How can microelectronics be used in other ways to put the *smart* in SPPE?

Question 1 has been answered in Chapter 3: the correct UV dose is applied by using a negative feedback control loop. Question 2 has been answered in Chapter 4: the reference value is updated in real-time based on the temperature and relative humidity. Question 3 is answered partly in Chapter 5: the filter quality monitoring is an extra feature of the submodule that demonstrates the possibilities of smart personal protective equipment. Question 3 is also answered partly in the following section on recommendations for new features of smart personal protective equipment.

To conclude, the Sensing and Control submodule satisfies the requirements, and the main questions have been answered. As far as we can conclude without using a physical prototype, the submodule operates correctly and can be used in the SPPE design. With this submodule, we have designed the next step in the direction of wearable UVGI devices, and thus one of the first steps in the direction of the future of face masks.

7.2 Discussion

In this section, several design choices of the Sensing and Control submodule are discussed further in depth.

7.2.1 Control signal modulation

In Chapter 4 the reference function $I_{\min}(T, RH)$ was discussed. The result of Equation (4.2) is the reference value $r(t)$ of the control loop of Chapter 3. By deriving Equation (4.2) in Section 4.2, we actually made an implicit design choice.
Returning to Equation (4.1), the product kD needs to remain constant to ensure mathematical sterilization of the filter material. By Equation (3.1), $D = It$, hence the product of kIt needs to remain constant. In Section 4.2 it was implicitly assumed that t is a given constant, $t = t_0$, and that I must be adjusted to the changing value of $k(T, RH)$. This is called *intensity modulation*.

However, it is also possible to assume I to be a given constant, $I = I_0$, and to adjust t to the changing value of $k(T, RH)$. This is called *time modulation*. This type of modulation actually provides two advantages over intensity modulation:

> When time modulation is used, the UVC LEDs are simply only turned on for a limited time, instead of using PWM to modify I. This greatly simplifies the UVC LED driver circuit of the UVGI submodule. Besides this, the modulation can cover a larger range. The design requirement of intensity modulation demanded that $I(T, RH) = (1 \pm 0.30)I_0$, in order to protect the UVC LEDs and their driver circuit. When time modulation is used, the UVC LEDs are simply turned on for some time, without having any constraints of the used components.

The options of modulation were only realized at a later stage in the project. Hence, the decision was made to not change the presented design of the reference function. The implementation of the time modulation in the reference function is recommended to further improve the performance of the SPPE. Other recommendations are given in Section 7.3.

7.2.2 Absolute pressure sensor

In Chapter 5 the absolute pressure sensor was introduced to calculate the density of air in Equation (5.4). The SPPE will often be used at atmospheric pressure, i.e. P_{abs}=101325 Pa. Is the pressure sensor necessary or can the atmospheric pressure simply be assumed?

The air pressure decreases if the altitude increases. The pressure is around 100 kPa at sea level, but it already decreases to 80 kPa at 2000 meter altitude. It is possible that the SPPE is used at this altitude. The difference in pressure is significant on Equation (5.4):

$$Q(P_{abs} = 100\text{kPa}) = 8.485\sqrt{\frac{\Delta P}{\rho_{air}(T, RH, P_{abs})}} = 8.485\sqrt{\frac{\Delta P}{1.1836}} = 55.44 \ \left[\frac{\text{L}}{\text{min}}\right]$$

$$Q(P_{abs} = 80\text{kPa}) = 8.485\sqrt{\frac{\Delta P}{\rho_{air}(T, RH, P_{abs})}} = 8.485\sqrt{\frac{\Delta P}{0.9457}} = 62.03 \ \left[\frac{\text{L}}{\text{min}}\right]$$

In both calculations it was assumed that T = 20°C, RH = 50%, ΔP = 50 Pa. Clearly, if the air pressure is assumed to be 100 kPa at all times, but it would actually decrease by 20 kPa, the result would be 10.6% off. This error is considered to be significant, and thus the absolute pressure sensor is added to avoid this error.

Of course, one could argue that the user of the SPPE will not instantly change from using the device at sea level to using it at 2000 meter altitude. However, if the SPPE is used on an aircraft where rapid pressure changes occur, the absolute pressure sensor is essential. Therefore, the SPPE is designed to be as self-contained as possible, and the absolute pressure sensor is included in the Sensing and Control submodule.

7.2.3 Differential pressure peak detection

Currently, the peak detection algorithm checks if the pressure threshold is violated during one minute. If this is the case, it is checked once more during one minute. A second violation of the threshold results in the indication signal getting set high, and the user is notified about the clogged mask.
The second set of measurements is meant to distinguish between a clogged filter and an occasional increase in pressure (sneezing, deep breathing, and so on). However, this technique is also not ideal. Perhaps the user simply sneezes twice: once at the end of the first set, and once at the beginning of the second set of measurements.

A better algorithm can be developed by counting the peaks of the breathing pattern, opposed to just keeping track of the maximum peak. If fifty peaks are detected that violate the threshold, that will almost certainly indicate a clogged filter. The improved algorithm will help improve the behaviour of the SPPE and it will lead to less filter material getting wasted.

7.2.4 Complete face mask design

An important design choice was to design the SPPE as a filter module, instead of as a complete face mask. The advantage of the filter module design is the modularity of the module, i.e. if a part breaks down, it is easier and less costly to replace.
However, if the SPPE is designed as a complete face mask, several components can be saved as they occur double. These include the microcontroller, the I^2C MUX, the battery, and the battery management system. This will, in turn, reduce the costs and the weight of the resulting face mask. Furthermore, the components can be placed closer to the user's face, thereby reducing the force that is exerted on the straps of the face mask.

7.2.5 Temperature and relative humidity sensors

In Section 4.3.1, the Si7021 temperature and relative humidity sensor was selected. It was preferred over the SHT35 because of the lower current consumption and the lower price, see Appendix B.2. Even though one of the advantages of the SHT35 was the plurality of I^2C addresses.
However, using the Si7021 requests that an I^2C MUX is used, such as the TCA9548A. This component adds to the total costs of the system. The combination of two Si7021 sensors and the I^2C MUX costs approximately the same, as the two SHT35 sensors.

When the sensor was selected, the SPPE project goal was to create a complete face mask with four sensors (as mentioned in Subsection 7.2.4). The cost difference between four Si7021 sensors and four SHT35 sensors is more significant of course, so the decision was made to use the Si7021 with an I^2C MUX. In hindsight, it would be best to use the SHT35, as this saves space on the control PCB, and the price is approximately the same.

7.2.6 Heating element

As was seen in Chapter 4, the viability of viruses drops if the temperature increases. Hence, if the temperature in the SPPE is increased, the inactivation of viruses is easier and requires less UV light. The temperature can be increased artificially by including a heating element in the SPPE. Such an element consists of a conductor that is subject to Ohmic heating. Most often, Positive Temperature Coefficient (PTC) heaters are used
We investigated the possibility of including such a heating element in the Sensing and Control submodule. However, three problems were encountered:

- The form factor. The element needs to be small in order to fit within the SPPE and not block the airflow. Also, it can not be included in the control system PCB, as the heat will tamper with the measurements of the other electronics.
- The power consumption. All of the considered heating elements had power requirements much higher than possible in the Sensing and Control submodule. All require a voltage of at least 12 V, and a current of at least 200 mA.
- The user's safety. The goal of the heating element is to increase the safety of the user by inactivating more harmful pathogens. However, the heating element will reach temperatures of almost 50 °C. The heating element will be too close to the user's face, and will cause extensive burnings. The heat will still be too intense even when the heating is only turned on during exhalation, as the heat lingers after the element is turned off.

For these reasons, we decided to not include the heating element in the Sensing and Control submodule.

7.2.7 Submodule pricing

Throughout this thesis, the emphasis was placed on designing the Sensing and Control submodule as a technology demonstrator. However, if the SPPE is further developed as an industrial product, its price will matter too. In this thesis, the selected components have been judged on their prices, just as well as their technical characteristics.
If the submodule is produced, some parts may be left out in order to save money. The following list selects the components that can be left out of the submodule, in ascending order of importance for the submodule. Please note that for every component that is left out, functionalities of the submodule will be lost.

1. The absolute pressure sensor is the first component to be left out. As discussed, this sensor is included to enlarge the self-contained nature of the SPPE. The SPPE will need to be manually reprogrammed for new pressure conditions in case the absolute pressure sensor is left out.
2. The differential pressure sensor and the rest of the filter quality function are the second set of components that will be left out. The main goal of this thesis was to include UVGI control in the SPPE. Hence, the control is prioritized over the filter quality function.
3. The temperature and relative humidity sensor is the third component to be left out. The reference function is left out too, and the control loop is used with a constant reference value. Hereby, the UVC LEDs are used less power-efficient.

The control loop and the UV sensor can never be left out of the SPPE.

7.3 Recommendations and future extensions

The current design of the Sensing and Control submodule can of course be extended to introduce new functionalities or improve the current ones. Below follows a list of recommendations for new or improved functionalities that we encourage future researchers and developers to explore.

- **Adaptive PID control** In this thesis, a PID controller was used. However, the electrical components start to drift after prolonged usage, and thus the control parameters need to be adjusted. In Section 1.3, the technique of Adaptive PID control was already mentioned. The techniques in the article of Alexandrov and Palenov [16] can be further analysed to judge its possible application in the SPPE's microcontroller.
- **Continuously radiate while exhaling.** If the user has been diagnosed with infectious pathogens, the SPPE can be programmed to radiate every time the user exhales. Hereby, the pathogens are inactivated and the infectivity of the user is reduced. In case the pathogens spread via the air, this functionality may prevent the need for self-isolation.
- **Climate control** The Sensing and Control submodule already measures a variety of ambient conditions, such as the temperature and the relative humidity in the face mask. If these values become too high or low a step toward active compensation can be made. Examples are cooling elements or dehumidifiers. making the SPPE climate controlled.

- **State-of-the-art sensors** In Section 1.3 the state of the art was discussed. Several state-of-the-art sensors were mentioned, flexible temperature sensors being an example. The sensors currently implemented by the Sensing and Control submodule can be replaced by this kind of specialised sensors allowing for a more compact design.
- **Air flow sensor** If it turns out that the airflow calculation of Equation (5.3) are not accurate enough, a dedicated airflow sensor can be implemented in the Sensing and Control submodule.
- **Sleep mode** Currently the ATmega328P is put into sleep mode by using the Watchdog Timer, as discussed in Section 6.2.2. However, this is not the most power-efficient way. A better result is achieved by using an interrupt pin to wake the microcontroller.
- **Microcontroller combination** The On-Board Power Management submodule also makes use of an ATmega328P. Perhaps the algorithms can be designed such that they can be combined into one algorithm, and one ATmega328P can be subtracted from the design. This will result in better power efficiency.
- **Extended reference function** The air flow is calculated in the filter quality function. For this, the air flow velocity is also calculated. As was seen in Figure 4.1, the air flow velocity has a negative effect on the UV efficiency. Hence, the reference function of Chapter 4 can be extended to also account for the changes in air flow velocity.
- **Aesthetics** The aesthetics of the face masks can be improved. Making the product a fashion statement. Also electronically the aesthetics can be improved. It is perhaps possible to indicate the user's facial expressions on the outside of the face mask. And if the SPPE is designed as a complete face mask. For example, the indication RGB LED can be placed more tactically.
- **Other Sensing and Control applications** Lastly, we would like to emphasize that the application of the Sensing and Control submodule is not limited to the SPPE. The possible further applications of the submodule include, but are not limited by:
  - other face masks, such as chemical respirators and firefighter masks.
  - stationary medical equipment, for example, products as the sterilization cabinets or locations that are difficult to reach.
  - air filters where UVGI is not applied. The filter quality function is useful for these applications too.
  - applications where additional sensors are required, for example, Geiger counters or gas detectors. By adding new sensors, more applications can benefit from the technologies of the Sensing and Control submodule.

7 | Bibliography

[1] M.-W. Wang, M.-Y. Zhou, G.-H. Ji, L. Ye, Y.-R. Cheng, Z.-H. Feng, J. Chen, "Mask crisis during the COVID-19 outbreak," *European Review for Medical and Pharmacological Sciences*, vol. 24, no. 6, pp. 3397–3399, 2020.

[2] D. Wang, M. Zhou, X. Nie, W. Qiu, M. Yang, X. Wang, T. Xu, Z. Ye, X. Feng, Y. Xiao, W. Chen, "Epidemiological characteristics and transmission model of Corona Virus Disease 2019 in China," *Journal of infection*, vol. 80, no. 5, pp. 25–27, 2020.

[3] W. Kowalsky, *Ultraviolet Germicidal Irradiation Handbook: UVGI for Air and Surface Disinfection*. Berlin, Germany: Springer, 2009.

[4] BioShift. "BioShift® Pass-Through UV-C Chamber – Large". [Accessed: Jun. 15, 2020]. [Online]. Available: https://www.once.lighting/product/bioshift-large/

[5] Germicidallamp. "UV4 AIR MOBIL 30 UV-C Lamp". [Accessed: Jun. 15, 2020]. [Online]. Available: https://www.germicidallamp.nl/webshop/producten/detail/36/uv4-air-mobil-30--uv-c-lamp.html?gclid=Cj0KCQjwuJz3BRDTARIsAMg-HxVzgqz8Yp6jUzpf2IP7gJUWsYKs0OGM2SjRn4PrSw_ZNO4PQWYXeVEaAh8pEALw_wcB

[6] Spectroline. "UVGI Sanitizing Cabinets". [Accessed: Jun. 15, 2020]. [Online]. Available: https://spectroline.com/germicidal-uv-cabinet/

[7] P.O. Nyangaresi, Y. Qin, G.L. Chen, B.P. Zhang, Y.H. Lu, L. Shen, "Comparison of the performance of pulsed and continuous UVC-LED irradiation in the inactivation of bacteria," *Water Research*, vol. 157, pp. 218–227, 2019.

[8] S.S. Nunayon, H.H. Zhang, A.C.K. Lai, "Comparison of disinfection performance of UVC-LED and conventional upper-room UVGI systems," *Indoor Air*, vol. 30, no. 1, pp. 180–191, 2019.

[9] G. Messina, M. Fattorini, S. Burgassi, M. Tani, Nante, G. Cevenini, "Novel UVC LED approach for disinfecting contact lenses," *European Journal of Public Health*, vol. 26, p. 459, 2016.

[10] W.Y. Zou, M. Sastry, J.J. Gooding, R. Ramanathan, V. Bansal, "Recent Advances and a Roadmap to Wearable UV Sensor Technologies," *Advanced Materials Technologies*, vol. 5, no. 4, p. 1901036, 2020.

[11] M. Hilal, J.I. Han, "Development of a Highly Flexible and Durable Fiber-Shaped Temperature Sensor Based on Graphene/Ni Double-Decked Layer for Wearable Devices," *IEEE Sensors Journal*, vol. 20, no. 10, pp. 5146–5154, 2020.

[12] F. Wang, J.J. Jiang, F.Q. Sun, L.F. Sun, T. Wang, Y.X. Liu, M.H. Li, "Flexible wearable graphene/alginate composite non-woven fabric temperature sensor with high sensitivity and anti-interference," *Cellulose*, vol. 27, no. 4, pp. 2369–2380, 2020.

[13] Y. Luo, Y.C. Pei, X.M. Feng, H. Zhang, B.H. Lu, L. Wang, "Silk fibroin based transparent and wearable humidity sensor for ultra-sensitive respiration monitoring," *Materials Letters*, vol. 260, p. 126945, 2020.

[14] J.H. Jang, J.I. Han, "Cylindrical relative humidity sensor based on poly-vinyl alcohol (PVA) for wearable computing devices with enhanced sensitivity," *Sensors and Actuators A-Physical*, vol. 261, pp. 268–273, 2017.

[15] T. Liu, C.Z. Zhu, W. Wu, K.N. Liao, "Facilely prepared layer-by-layer graphene membrane-based pressure sensor with high sensitivity and stability for smart wearable devices," *Journal of Materials Science and Technology*, vol. 45, pp. 241–247, 2020.

[16] A. Alexandrov and M. Palenov, “Adaptive pid controllers: State of the art and development prospects,” *Automation and Remote Control*, vol. 75, pp. 188–199, 2014.

[17] C.-C. Tseng, C.-S. Li, “Inactivation of viruses on surfaces by ultraviolet germicidal irradiation,” *Journal of Occupational and Environmental Hygiene*, vol. 4, no. 6, pp. 400–405, November 2007.

[18] M.E.R. Darnell, K. Subbarao, S.M. Feinstone, D.R. Taylor, “Inactivation of the Coronavirus that Induces Severe Acute Respiratory Syndrome, SARS-CoV,” *Journal of Virological Methods*, vol. 121, pp. 85–91, August 2004.

[19] D. Neamen, *Semiconductor Physics And Devices (international edition)*, 4th ed. USA: McGraw-Hill, Inc., 2012.

[20] T.V. Blank, Y.A. Goldberg, E.V. Kalinina, O.V. Konstantinov, “4H–SiC-Based Photodetector of Carcinogenic UV Radiation,” *Technical Physics*, vol. 53, no. 1, pp. 81–84, 2008.

[21] E.V. Kalinina, G.N. Violina, I.P. Nikitina, E.V. Ivanova, V.V. Zabrodski, M.H. Shvarts, S.A. Levina, V. Nikolaev, “Effect of temperature on the characteristics of 4H-SiC UV photodetectors,” *Semiconductors*, vol. 54, pp. 246–252, 2020.

[22] *GUVB-S11SD: Technical data*, Roithner Lasertechnik, 2011, "GUVB Datasheet". [Online]. Available: http://www.roithner-laser.com/datasheets/pd/uv/guvb-s11sd.pdf

[23] D. Li, K. Jiang, X. Sun, C. Guo, “AlGaN photonics: recent advances in materials and ultraviolet devices,” *Advances in Optics and Photonics*, vol. 10, no. 1, pp. 43–110, 2018.

[24] *SiC UV Photodiode Selection Guide*, Sglux, no date, "Photodiode Selection Guide". [Online]. Available: http://www.jinzon.com/pdf/UV_photodiode_selection_guide.pdf

[25] A. J. Montagne, *Structured Electronic Design, a conceptual approach to amplifier design*, 1st ed. The Netherlands, Delft: Delft Academic Press, 2020.

[26] *Single-supply, microPower CMOS operational amplifiers microAmplifier ™ Series*, Texas Instruments, 2005, "OPA336 Datasheet". [Online]. Available: https://www.ti.com/lit/ds/symlink/opa336.pdf?ts=1591861122492&ref_url=https://www.ti.com/product/OPA336

[27] G. F. Franklin, D. J. Powell, and A. Emami-Naeini, *Feedback Control of Dynamic Systems (Global Edition)*, 7th ed. England, Essex: Pearson, 2014.

[28] *Simulink*, MathWorks, 2020, [Accessed: Apr. 10, 2020]. [Online]. Available: https://www.mathworks.com/products/simulink.html

[29] PID Autotuning for a Plant Modeled in Simulink. Mathworks. (2020, June 2). [Online]. Available: https://nl.mathworks.com/help/slcontrol/ug/pid-autotuning-for-a-plant-modeled-in-simulink.html

[30] M. S. Fadali and A. Visioli, *Introduction to Digital Control*. Boston: Academic Press, 2009.

[31] Arduino web site. Arduino. [Accessed: Jun. 10, 2020]. [Online]. Available: https://www.arduino.cc/

[32] *Arduino IDE 1.8.13*, Arduino, 2020, [Accessed: Jun. 12, 2020]. [Online]. Available: https://www.arduino.cc/en/Main/Software

[33] Introduction to I^2C and SPI protocols. Byte paradigm. [Accessed: May 22, 2020]. [Online]. Available: https://www.byteparadigm.com/applications/introduction-to-i2c-and-spi-protocols/

[34] *"ATmega328P 8-bit AVR Microcontrollers"*, Atmel, 2016, "ATmega328P Datasheet". [Online]. Available: https://static1.squarespace.com/static/55abde31e4b0249b70d593c9/t/5a6b4209c830255e65ed23b8/1516978711075/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_Datasheet.pdf

[35] B. Beauregard. Arduino PID Library. [Accessed: Jun. 1, 2020]. [Online]. Available: http://brettbeauregard.com/blog/2011/04/improving-the-beginners-pid-introduction/

[36] S.W. Kim, M.A. Ramakrishnan, P.C. Raynor, S.M. Goyal, “Effects of humidity and other factors on the generation and sampling of a coronavirus aerosol,” *Aerobiologia*, vol. 23, pp. 239–248, 2007.

[37] A.J. Prussin II, D.O. Schwake, K. Lin, D.L. Gallagher, L. Buttling, L.C. Marr, "Survival of the Enveloped Virus Phi6 in Droplets as a Function of Relative Humidity, Absolute Humidity, and Temperature," *Applied and Environmental Microbiology*, vol. 84, no. 12, pp. e00 551–18, 2018.

[38] K. Lin, L.C. Marr, "Humidity-Dependent Decay of Viruses, but Not Bacteria, in Aerosols and Droplets Follows Disinfection Kinetics," *Environmental Science and Technology*, vol. 54, pp. 1024–1032, 2020.

[39] H. Zhang, X. Jin, S.S. Nunayon, A.C.K. Lai, "Disinfection by in-duct ultraviolet lamps under different environmental conditions in turbulent airflows," *Indoor Air.*, vol. 30, pp. 500–511, 2020.

[40] G. Byrns, B. Barham, L. Yang, K. Webster, G. Rutherford, G. Steiner, D. Petras, M. Scannell, "The uses and limitations of a hand-held germicidal ultraviolet wand for surface disinfection," *Journal of Occurpational and Environmental Hygiene*, vol. 14, no. 10, pp. 749–757, 2017.

[41] M. Ivanov, "Exhaled air speed measurements of respiratory air flow, generated by ten different human subjects, under uncontrolled conditions," *E3S Web of Conferences*, vol. 111, p. 02074, 2019.

[42] J.W. Tang, A.D. Nicolle, C.A. Klettner, J. Pantelic, L. Wang, A.B. Suhaimi, A. Y.L. Tan, G.W.X. Ong, R. Su, C. Sekhar, D.D.W. Cheong, K.W. Tham, "Airflow Dynamics of Human Jets: Sneezing and Breathing - Potential Sources of Infectious Aerosols," *PLoS One*, vol. 8, no. 4, p. e59970, 2013.

[43] D. W. Hendricks, *Water Treatment Unit Processes: Physical and Chemical*, 1st ed. USA: CRC Press, 2006.

[44] L.M. Casanova, S. Jeon, W.A. Rutala, D.J. Weber, M.D. Sobsey, "Effects of Air Temperature and Relative Humidity on Coronavirus Survival on Surfaces," *Applied and Environmental Microbiology*, vol. 76, no. 9, pp. 2712–2717, 2010.

[45] J. Hermann, S. Hoff, C.M.-Z., K.-J. Yoon, M. Roof, A. Burkhardt, J. Zimmerman, "Effect of temperature and relative humidity on the stability of infectious porcine reproductive and respiratory syndrome virus in aerosols," *Vet. res.*, vol. 38, pp. 81–93, 2007.

[46] K.H. Chan, J.S. Malik Peiris, S.Y. Lam, L.L. M. Poon, K.Y. Yuen, W.H. Seto, "The Effects of Temperature and Relative Humidity on the Viability of the SARS Coronavirus," *Advances in Virology*, pp. 1–7, 2011.

[47] M.-H. Woo, A. Grippin, D. Anwar, T. Smith, C.-Y. Wu, J.D. Wander, "Effects of Relative Humidity and Spraying Medium on UV Decontamination of Filters Loaded with Viral Aerosols," *Applied and Environmental Microbiology*, vol. 78, no. 16, pp. 5781–5787, 2012.

[48] T.D. Cutler, C. Wang, S.J. Hoff, J.J. Zimmerman, "Effect of temperature and relative humidity on ultraviolet (UV254) inactivation of airborne porcine respiratory and reproductive syndrome virus," *Veterinary microbiology*, vol. 159, pp. 47–52, 2012.

[49] *I^2C Humidity and temperature sensor*, Silicon Labs, 2016, "Si7021-A20 Datasheet". [Online]. Available: https://www.silabs.com/documents/public/data-sheets/Si7021-A20.pdf

[50] *Digital Humidity Sensor*, Innovative Sensor Technology, 2012, "HYT-939 Datasheet". [Online]. Available: http://www.farnell.com/datasheets/1643980.pdf?_ga=2.205831763.1936061804.1591366836-1875123382.1584111373

[51] *Capacitive Humidity Sensor*, Innovative Sensor Technology, Unknown, "P14 FemtoCap Datasheet". [Online]. Available: https://www.ist-ag.com/sites/default/files/DHP14-FemtoCap_E.pdf

[52] *Humidity and Temperature Sensor*, Amph, 2018, "ChipCap 2-SPI Datasheet". [Online]. Available: https://www.amphenol-sensors.com/en/telaire/humidity/527-humidity-sensors/3363-chipcap-2-sip

[53] *SHT3x DIS Data Sheet: Humidity and Temperature Sensor*, Sensirion, 2011, "SHT35T-DIS-F Datasheet". [Online]. Available: https://www.sensirion.com/fileadmin/user_upload/customers/sensirion/Dokumente/2_Humidity_Sensors/Datasheets/Sensirion_Humidity_Sensors_SHT3x_Datasheet_digital.pdf

[54] *Low-Voltage 8-Channel I^2C Switch with reset*, Texas Instruments, 2019, "TCA9548A Datasheet". [Online]. Available: http://www.ti.com/lit/ds/symlink/tca9548a.pdf?ts=1591367857522

[55] I^2C Design Mathematics: Capacitance and Resistance. AllAboutCircuits. [Accessed: Jun. 5, 2020]. [Online]. Available: https://www.allaboutcircuits.com/technical-articles/i2c-design-mathematics-capacitance-and-resistance/

[56] ElectronicsNotes. Standard Resistor Values: E3, E6, E12, E24, E48, and E96. [Accessed: Jun. 18, 2020]. [Online]. Available: https://www.electronics-notes.com/articles/electronic_components/resistors/standard-resistor-values-e-series-e3-e6-e12-e24-e48-e96.php

[57] *Humidity and temperature sensor designer's guide*, Silicon Labs, 2016, "Si7021 Designer's guide". [Online]. Available: https://www.silabs.com/documents/public/application-notes/AN607.pdf

[58] *SHTxx and STSxx Design guide*, Sensirion, 2019, "SHT35 Designer's guide". [Online]. Available: https://www.sensirion.com/fileadmin/user_upload/customers/sensirion/Dokumente/2_Humidity_Sensors/Application_Note/Sensirion_Humidity_Sensors_Design_Guide.pdf

[59] H. Jung, J. Kim, S. Lee, J. Lee, J. Kim, P. Tsai, and C. Yoon, "Comparison of filtration efficiency and pressure drop in anti-yellow sand masks, quarantine masks, medical masks, general masks, and handkerchiefs," *Aerosol and Air Quality Research*, vol. 14, pp. 991–1002, 2014.

[60] *Comparison of FFP2, KN95, and N95 and Other Filtering Facepiece Respirator Classes*, 3M, 2020. [Online]. Available: https://multimedia.3m.com/mws/media/1791500O/comparison-ffp2-kn95-n95-filtering-facepiece-respirator-classes-tb.pdf

[61] M. J. Matteson and C. Orr, *Filtration: Principles and Practices, Second Edition, Revised and Expanded (Chemical Industries)*, 2nd ed. USA: CRC Press, 1986.

[62] Q. Wang, B. Maze, H. V. Tafreshi, B. Pourdeyhimi, "On the pressure drop modeling of monofilament-woven fabrics," *Chemical Engineering Science*, vol. 62, no. 17, pp. 4817–4821, 2007.

[63] A. Picard, R. S. Davis, M. Gläser, K. Fujii, "Revised formula for the density of moist air (CIPM-2007)," *Metrologia*, vol. 45, pp. 149–155, 2008.

[64] *"Datasheet SDP3x Digital Differential Pressure Sensor"*, Sensirion, 2017, "SDP3x Datasheet". [Online]. Available: https://www.sensirion.com/fileadmin/user_upload/customers/sensirion/Dokumente/8_Differential_Pressure/Datasheets/Sensirion_Differential_Pressure_Sensors_SDP3x_Digital_Datasheet.pdf

[65] *"ABP Series Basic Board Mount Pressure Sensors"*, Honeywell, 2019, "ABPMRRV001PD2A3 Datasheet". [Online]. Available: https://sensing.honeywell.com/honeywell-sensing-basic-board-mount-pressure-abp-series-datasheet-32305128.pdf

[66] *"AMEMS pressure sensor"*, ST, 2017, "LPS33HWTR Datasheet". [Online]. Available: https://www.st.com/resource/en/datasheet/lps33hw.pdf

[67] hoffmannjan, helmut64, ChuckVanzant,ladyada, siddacious, dherrada, caternuson, stickbreaker, johnduhart, dastels, *Arduino library for Adafruit Si7021, Release 1.3.0*, Adafruit, 2020, [Accessed: Jun. 2, 2020]. [Online]. Available: https://github.com/adafruit/Adafruit_Si7021

[68] siddacious, ladyada, dherrada, hoffmannjan, *Adafruit LPS35HW, Release 1.0.4*, Adafruit, 2020, [Accessed: Jun. 2, 2020]. [Online]. Available: https://github.com/adafruit/Adafruit_LPS35HW

[69] ij96, *Honeywell ABP library, Release 1.0.0*, 2020, [Accessed: Jun. 3, 2020]. [Online]. Available: https://github.com/ij96/Honeywell_ABP

[70] TheKurks. Tutorial: a guide to putting your Arduino to sleep. [Accessed: Jun. 12, 2020]. [Online]. Available: https://thekurks.net/blog/2018/1/24/guide-to-arduino-sleep-mode

[71] *"AN5063 Application note LPS33HW guidelines for system integration"*, ST, 2017, "LPSHW Application note". [Online]. Available: https://www.st.com/resource/en/application_note/dm00420205-lps33hw-digital-pressure-sensor-guidelines-for-system-integration-stmicroelectronics.pdf

A | UVGI control

A.1 UV sensor choice

Table A.1: The UV photodiode comparison based on: model, price, material, spectral efficiency at 270 nm and 280 nm, and sensitivity. From this table the GUVB-S11SD is selected for SPPE.

Model	**Price**	**Material**	**Spectral efficiency (at 270nm) - (at 280nm)**	**Responsivity**
TOCON_Cx	€ 160,-	SiC based	(0.9) (0.8)	Multiple options
GUVV-T10GD	€ 15,-	InGaN	(0.54) (0.6)	30 mw/cm^2
SG01S-Cx	€ 80,-	SiC based	(0.98) (0.82)	Multiple options
SIC01S-18	€ 25,-	SiC	(0.9) (0.99)	10 mW/cm^2 → 645 nA
GUVB-S11SD	€ 3,80	AlGaN	(0.9) (0.95)	10 mW/cm^2 → 690 nA

B | Reference function

B.1 Matlab script for I_{min}(T, RH)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Script to compute the inactivation constant (k) and minimum intensity (I)
% in order to inactivate virusses, bacteria or fungi. This doses is needed
% to mathematically sterilize the species, i.e. only let one-millionth of
% the population survive.
% Author: M. Goddijn
% Date: 14-05-2020
% Inputs: T range and RH range, Kowalsky inactivation constant k0, time during
% which the intensity is being applied t0, required fraction of survival S.
% Outputs: Surface plots of k and I_min.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
close all;
clear variables;

S0 = 10^(-6);  % Only one-millionth of the population survives
t0 = 0.5;      % [s]
k0 = 0.377;    % [m^2/J] coronavirus ssRNA in air (Kowalsky, 2008)

T_meas = [0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50];
RH_meas = [0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100];

% Create the inactivation constant matrix
% Columns represent constant RH
% Rows represent constant T
k = k0.*[
     0.70, 0.70, 0.75, 0.75, 0.80,   0.80, 0.80, 0.80, 0.75, 0.75, 0.70; % 0
     0.70, 0.70, 0.75, 0.75, 0.80,   0.85, 0.85, 0.85, 0.85, 0.75, 0.70; % 5
     0.75, 0.75, 0.80, 0.80, 0.85,   0.90, 0.95, 0.95, 0.95, 0.85, 0.80; % 10
     0.78, 0.79, 0.80, 0.85, 0.90,   0.95, 0.95, 1.00, 1.00, 0.90, 0.80; % 15
     0.85, 0.86, 0.88, 0.90, 0.95,   1.00, 1.05, 1.13, 1.15, 1.13, 0.95; % 20

     1.00, 1.00, 1.03, 1.06, 1.09,   1.10, 1.12, 1.15, 1.17, 1.15, 1.00; % 25
     1.08, 1.10, 1.12, 1.13, 1.14,   1.15, 1.17, 1.18, 1.18, 1.17, 1.08; % 30
     1.10, 1.13, 1.15, 1.15, 1.16,   1.17, 1.19, 1.20, 1.20, 1.19, 1.15; % 35
     1.13, 1.15, 1.17, 1.17, 1.20,   1.23, 1.24, 1.25, 1.26, 1.23, 1.18; % 40
     1.15, 1.16, 1.18, 1.21, 1.23,   1.27, 1.27, 1.28, 1.28, 1.26, 1.20; % 45
     1.16, 1.17, 1.20, 1.23, 1.25,   1.28, 1.29, 1.30, 1.30, 1.30, 1.22; % 50
];

% Compute the minimum dose and minimum intensity
D_min = -log(S0)./k;  % [J/m^2]
I_min = D_min /t0;    % [W/m^2]

% Create a surface plot to illustrate k
figure(1)
s1 = surf(RH_meas, T_meas, k, 'FaceAlpha', 0.8);
s1.EdgeColor = 'none';
```

```
colorbar;
xlabel('RH [%]');
ylabel('T [*C]');
zlabel('k [m^2/J]');

% Create a surface plot to illustrate the minimum D
figure(2)
s1 = surf(RH_meas, T_meas, I_min, 'FaceAlpha', 0.8);
s1.EdgeColor = 'none';
colorbar;
xlabel('RH [%]');
ylabel('T [*C]');
zlabel('I_{min} [W/m^2]');


% End of file
```

B.2 Humidity and temperature sensor comparison

The comparison between several temperature and relative humidity sensors is shown in Table B.1. From this table the SHT3x-DIS and the Si7021-A20 are deemed the most promising.

Table B.1: The temperature (T) and relative humidity (RH) sensor comparison based on: price, range, accuracy, output communication, and response time.

Model	**Price**	**RH range and accuracy**	**T range and accuracy**	**Output**	**Response time**	**Others**
HYT-939	€ 56	[0, 100]%, ±1.8%	-	I^2C	10s	V∈ [2.7, 5.5]
P14 Femtocap-G	€ 12,44	[0, 100]%, 0.3pF / 1%	-	Analog	<3s	SMD
HPP801A031	€ 5,39	[0, 100]%, 0.31pF / 1%	-	Analog	-	Internal circuit, bulky
AHT20	€ 4,90	[0, 100]%, ±2%	[-40, +85] °°C ±0.3°C	I^2C	8s	Complete datasheet
ChipCap 2-SIP	€ 20,16	[20, 80]%, ±3%	[-40, +125] °C ±0.3°C	I^2C	7s	V∈ [3.3, 5.5]
Si7021-A20	€ 3,18	[0, 100]%, ±3%	[-10, +85] °C ±0.4°C	I^2C	0.7s (T) 18s (RH)	Excellent documentation, SMD
SHT3x-DIS	€ 4,70 (SHT31)	[0, 100]%, ±2%	[-40, +120] °C ±0.3°C	I^2C	2s (T) 8s (RH)	Excellent documentation, SMD
DHT22	€ 4,50	[0, 100]%, ±5%	[-40, +80] °C	Analog	-	No official datasheet

B.3 Comparison between Si7021-A20 and the HT35-DIS

Based on the comparison in Table B.2, the Si7021-A20 (Si7021 in short) temperature and humidity sensor is selected. This sensor is cheaper than the SHT35, and it draws less current (and is thus more power efficient). Both sensors have excellent documentation, are protected against UV light and water, and are applicable in the specified ranges for T and RH. The advantages of the SHT35 (slightly more accurate, tiny bit smaller, and two possible I^2C addresses) are not deemed to be enough to overcome the advantages of the Si7021. Hence, the Si7021 is the sensor that will be used in the SPPE.

Table B.2: A more detailed comparison between the Si7021-A20 and the SHT3T-DIS-F temperature and humidity sensors. Green indicates a sensor is better than the other, orange indicates they are equal, and red indicates a sensor is worse than the other.

Si7021-A20-GM1	**SHT35-DIS-F**
Protection cover - Teflon (PTFE) - IP67 - UV resistant - Cleaning agent sensitive (drift)	Protection cover - Teflon (PTFE) - IP67 - UV resistant - Cleaning agent sensitive (drift)
€ 3.03 (Mouser electronics) € 3.81 (Farnell)	€ 9.05 (Mouser electronics) € 10.63 (Farnell)
Applicable in respiratory devices	Applicable in respiratory devices
Long-term drift: - < 0.01 °C/yr - < 0.25 %RH/yr	Long-term drift: - < 0.03 °C/yr - < 0.25 %RH/yr
Current consumption: - 0.06 μA stand-by - 150/90 μA measuring - 3500 μA during I2C	Current consumption: - 0.2 μA stand-by - 600 μA measuring - *unknown* μA during I2C
Hysteresis: - 1% RH	Hysteresis: - 0.8% RH
Documentation is very good: datasheet and user guide available.	Documentation is very good: datasheet and user guide available.r
Dimensions (l, w, h): 3, 3, 0.7 mm	Dimensions (l, w, h): 2.5, 2.5, 0.9 mm
I2C address: 0x40	I2C address: 0x44 or 0x45 (by connecting one pin)
Accuracy: ∓ 0.4 °C, ∓ 3% RH	Accuracy: ∓ 0.3 °C, ∓ 2% RH

C | Filter Quality

C.1 Differential pressure sensor comparison

The comparison between several differential pressure sensors is given in Table C.1. The accuracy of very little sensors are given. The ABPMRRV001PD2A3 is selected to be used in the SPPE, due to its I^2C possibilities, and its good price-sensitivity ratio (opposed to the HSCDRRV001PD2A3). From this table the ABPMRRV001PD2A3 is selected to be used in the SPPE.

Table C.1: The differential pressure sensor comparison based on: price, type, minimum pressure P_min, maximum pressure P_max, sensitivity, and others.

Model	Price	Type	P_min	P_max	Sensitivity	Others
MP3V5004DP	€ 13,20	Differential	0 Pa	3920 Pa	0.6V/kPa	Small sensor 3.3 V
BPS310	€ 11,72	Differential	0 Pa	1034 Pa	-	Small
MPXV7002 Series	€ 10,112	Differential	0 Pa	2 kPa	1V/kPa	5V
SDX01D4	€ 72,77	Differential	0 Pa	6 Pa	-	20 V, $t_{resp.}$ 100 μs
LMIS500U	€ 83,97	Differential	0 Pa	500 Pa	-	Extensive documentation, 3.3 V, [0,97]% RH
SDP31	€ 30,69	Differential	0 Pa	500 Pa	-	I^2C, 3.3V. Small.
MPR series	€ 6,35	Absolute	0 Pa	103420 Pa	-	I^2C, many variations
ABPMLNV006KG2A3	€ 14,06	Gauge	0 Pa	6000 Pa	-	I^2C
HSCDRRV001PD2A3	€ 36 - 49	Differential	0 Pa	1 psi	1.46 Pa (accuracy)	[3.3, 12] V
5525DSO-DB001DS	€ 20,39	Differential	0 Pa	6.89 kPa	-	3.3 V, I^2C
BPS125-AD0P15-2DG	€ 17,73	Differential	-	1 kPa	-	3.3 V, I^2C, not small.
ABPMRRV001PD2A3	€ 11.82	Differential	0 Pa	1 psi	17 Pa (accuracy)	3.3V, I^2C

C.2 Absolute pressure sensor comparison

The comparison between several absolute pressure sensors is given in Table C.2. Both the documents of the LPS35HWTR and the LPS33HWTR specify their resistance against water. Based on the excellent documentation, the LPS33HWTR is selected, even though it is slightly more expensive.

Table C.2: The absolute pressure sensor comparison based on: price, type, minimum pressure p_min, maximum pressure p_max, accuracy, and others.

Model	Price	Type	P_min	P_max	Accuracy	Other
LPS35HWTR	€ 3.16	Absolute	260 hPa	1260 hPa	±100 Pa	Small sensor, 3.3 V, watertight
BMP180	€ 8,85	Absolute	300 hPa	1100 hPa	± 30 Pa	Small, 3.3 V
MPL3115A2	€ 5,65	Absolute	200 hPa	1100 hPa	± 50 Pa	medium accuracy
LPS33HWTR	€ 5,65	Absolute	260 hPa	1260 hPa	±100 Pa	Watertight, 3.3 V, very detailed notes.

C.3 Absolute pressure sensor placement

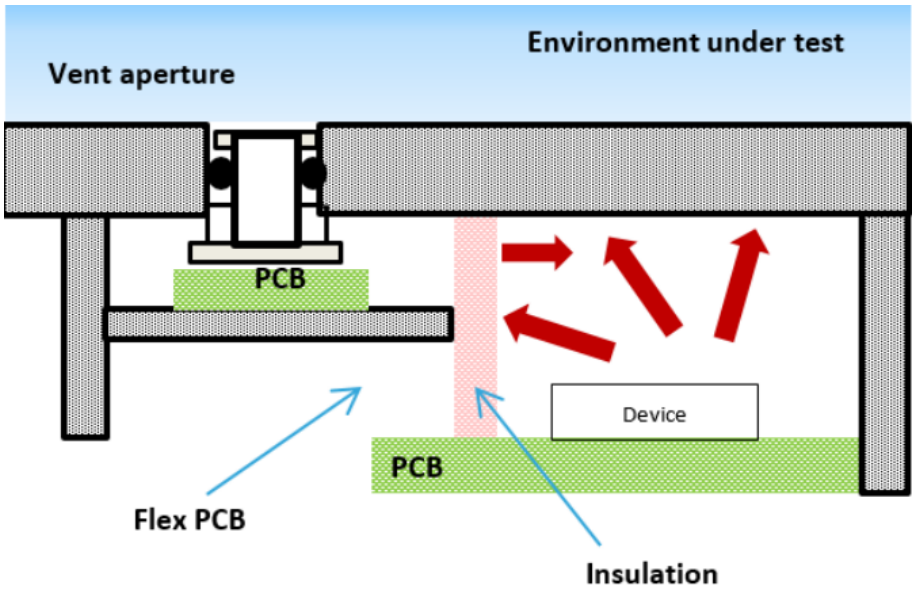
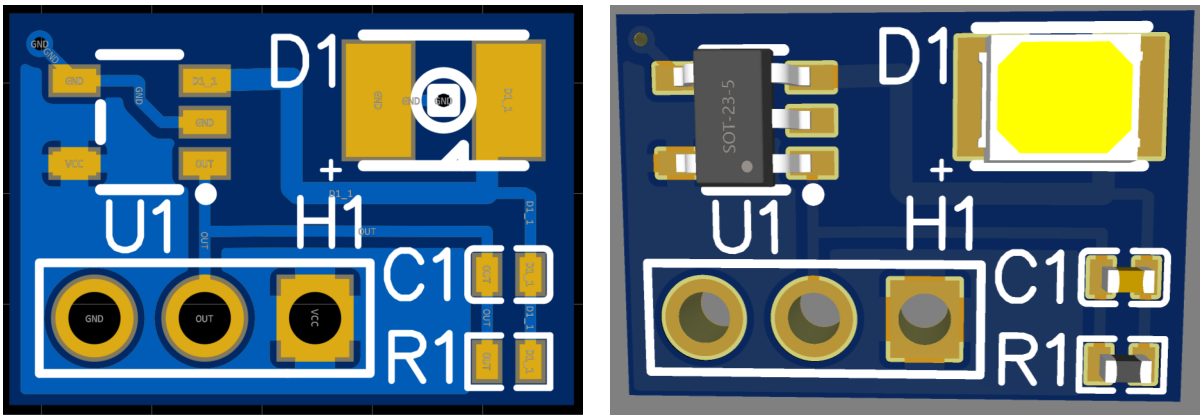


Figure C.1: The best placement of the LPS33HWTR absolute pressure sensor [71]. The image illustrates how the sensor would be connected through the side of the SPPE.

D | Sensing and Control finalization

D.1 Printed circuit board designs

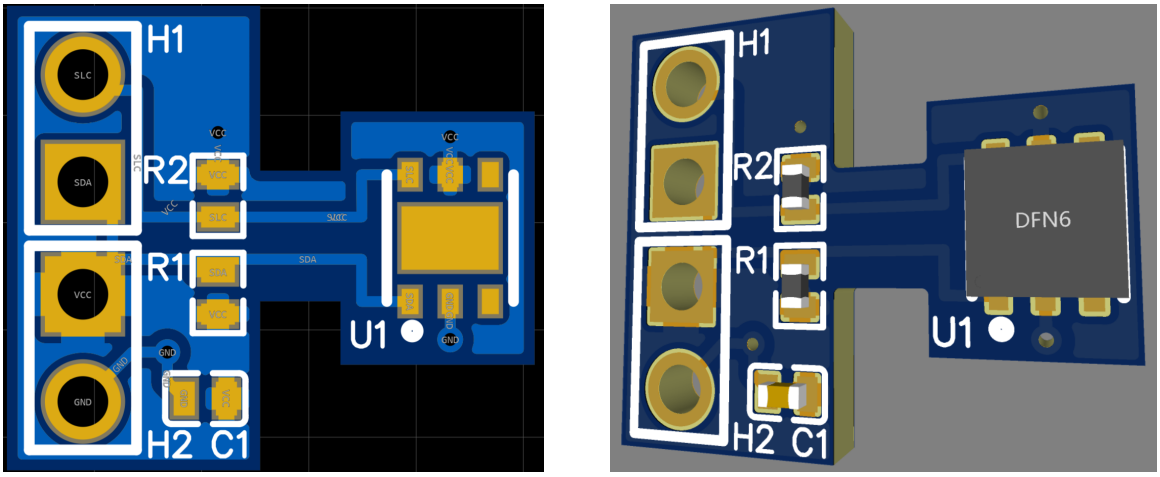
Figures D.1-D.3 show the printed circuit board designs of the UV sensor, the temperature and relative humidity (T+RH) sensor, and the absolute pressure sensor, respectively.



(a) PCB design of the UV sensor.

(b) Three-dimensional view of the UV sensor.

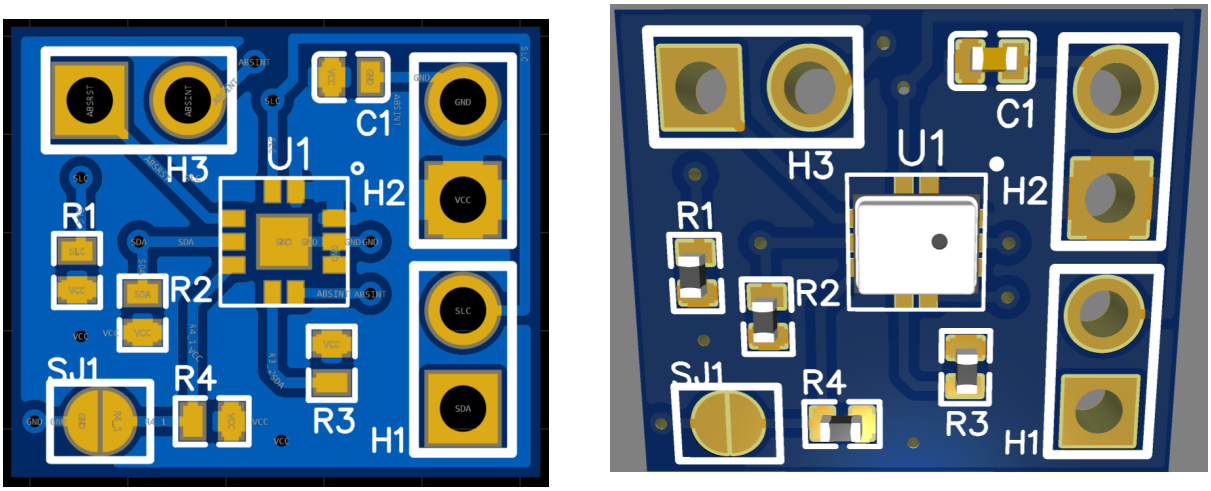
Figure D.1: In-depth design of the UV sensor. The size of the PCB is 13x9x5 mm, including components.



(a) PCB design of the T+RH sensor.

(b) Three-dimensional view of the T+RH sensor.

Figure D.2: In-depth design of the temperature and relative humidity sensor. The size of the PCB is 13x11x6 mm, including components.



(a) PCB design of the absolute pressure sensor.

(b) Three-dimensional view of the absolute pressure sensor.

Figure D.3: In-depth design of the absolute pressure sensor. The size of the PCB is 14x12x6 mm, including components.

D.2 Complete algorithm

8

```
/*
  DISCLAIMER
  This programme makes use of libraries developed by other authors. These are:
    - Adafruit SleepyDog Library 1.2.0,   by Adafruit <info@adafruit.com>
    - Adafruit Si7021 Library 1.3.0,      by Adafruit <info@adafruit.com>
    - Adafruit LPS35HW Library 1.0.4,     by Adafruit <info@adafruit.com>
    - Honeywell ABP Library 1.0.0,        by ij96
    - Arduino PID Library 1.2.0,          by Brett Beauregard <br3ttb@gmail.com>
  Please see the copyright statement of Adafruit at the end of this file.

  SPPE Sensing and Control
  Authors: Jasper-Jan Lut and Michael Goddijn
  Date: 02-06-2020
  Description: This programme is the implementation of the SPPE's Sensing and Control
      subsystem.
               The programme is operational and can be uploaded on the Arduino Pro Mini.
               The programme measures the sensor outputs and contains several functions.
               The programme's output is a signal to the LED arrays indicating Iout, which is
               computed using PID control.
               The following sensors are used in the Sensing and Control subsystem:
                 - UV sensor: Roithner GUVB-S11SD photodiode             , analog operation
                 - Temperature + Relative Humidity sensor: Si7021-A20-GM1 , I2C communication
                 - Differential pressure sensor: ABPMRRV001PD2A3          , I2C communication
                 - Absolute pressure sensor: LPS33HWTR                    , I2C communication
               An I2C MUX is used, as the SPPE contains two UV sensors, two T+RH sensors, and
               two LED arrays. The TCA9548A is used for this I2C MUX.
               Even though an LPS33HWTR is used, the Adafruit LPS35HW Library can be used.
*/

// // Include libraries // //
  #include <math.h>                 // Used for simple math operations
  #include <Adafruit_SleepyDog.h>   // Used for the sleep mode
  #include <Adafruit_Si7021.h>      // T+RH sensor, library used for I2C communication
  #include <Adafruit_LPS35HW.h>     // Pabs sensor, library used for I2C communication
  #include <Honeywell_ABP.h>        // Pdiff sensor, library used for I2C communication
  #include <Wire.h>                 // Used for the I2C communication on the Arduino
  #include <PID_v1.h>               // Used for the PID control

// // Define the input and output pins // //
  const int UVsenspin_1    = 23;   // Input of UV sensor 1
  const int UVsenspin_2    = 24;   // Input of UV sensor 2
  const int UVLEDpin_1     = 9;   // Output to LED array 1, PWM
  const int UVLEDpin_2     = 10;  // Output to LED array 2, PWM
  const int LEDRpin        = 15;   // PWM pin
  const int LEDGpin        = 14;   // PWM pin
  const int LEDBpin        = 13;   // PWM pin

// // Initialize the sensors // //
  // TCA9548A I2C MUX intialization
  const int I2CMUXaddress   = 0x70;
  const int busSi7021_1     = 2;
  const int busSi7021_2     = 3;
  int selectSi7021;

  // LPS35HW sensor intialization
  Adafruit_LPS35HW lps33hwtr = Adafruit_LPS35HW();

  // Si7021 sensor intialization
  Adafruit_Si7021 Si7021 = Adafruit_Si7021();
  bool enableHeater           = false;    // The Si7021's heating element to evaporate moist
  bool errorSensorSi7021      = false;    // Used for debugging the sensor
  float tempMeasurement       = 25;       // [degree C], Average T value
  float humidityMeasurement   = 50;       // [%], Average RH value
  bool Si7021_debugMode       = true;     // Use this mode to debug in case of errors

  // ABPMRRV001PD2A3 sensor intialization
```

```
66   Honeywell_ABP ABP(0x28, 0, 1, "psi"); // I2C address, Pmin, Pmax, unit (is converted to Pa
         later)
67
68 // // PID controller setup // //
69   double PIDref, PIDin, PIDout;
70   double Kp = 77.6827;
71   double Ki = 6099.0266;
72   double Kd = 0.1838; // the negative sign is included in the PID.comute();
73   double PIDoutMin = 0;
74   double PIDoutMax = 221;
75   PID myPID(&PIDin, &PIDout, &PIDref, Kp, Ki, Kd, P_ON_E, DIRECT);
76
77
78 // // Define the Reference function's values // //
79   float k0  = 0.377;          // [m^2/J], Kowalksy inactivation constant
80   float S0  = 10^(-6);        // [-], Fraction of survival
81   float t0  = 105;            // [s], Inactivation time
82   float kVal, Dmin, Imin;
83   float kTable[11][11] = {
84          {0.70, 0.70, 0.75, 0.75, 0.80,   0.80, 0.80, 0.80, 0.75, 0.75, 0.70},  // 0   deg C
85          {0.70, 0.70, 0.75, 0.75, 0.80,   0.85, 0.85, 0.85, 0.85, 0.75, 0.70},  // 5   deg C
86          {0.75, 0.75, 0.80, 0.80, 0.85,   0.90, 0.95, 0.95, 0.95, 0.85, 0.80},  // 10  deg C
87          {0.78, 0.79, 0.80, 0.85, 0.90,   0.95, 0.95, 1.00, 1.00, 0.90, 0.80},  // 15  deg C
88          {0.85, 0.86, 0.88, 0.90, 0.95,   1.00, 1.05, 1.13, 1.15, 1.13, 0.95},  // 20  deg C
89
90          {1.00, 1.00, 1.03, 1.06, 1.09,   1.10, 1.12, 1.15, 1.17, 1.15, 1.00},  // 25  deg C
91          {1.08, 1.10, 1.12, 1.13, 1.14,   1.15, 1.17, 1.18, 1.18, 1.17, 1.08},  // 30  deg C
92          {1.10, 1.13, 1.15, 1.15, 1.16,   1.17, 1.19, 1.20, 1.20, 1.19, 1.15},  // 35  deg C
93          {1.13, 1.15, 1.17, 1.17, 1.20,   1.23, 1.24, 1.25, 1.26, 1.23, 1.18},  // 40  deg C
94          {1.15, 1.16, 1.18, 1.21, 1.23,   1.27, 1.27, 1.28, 1.28, 1.26, 1.20},  // 45  deg C
95          {1.16, 1.17, 1.20, 1.23, 1.25,   1.28, 1.29, 1.30, 1.30, 1.30, 1.22}   // 50  deg C
96        }; // Rows correspond with T, columns correspond with RH
97   float Tinterval   = 5;  // [degC] The step size between consecutive T elements
98   float RHinterval  = 10; // [%] The step size between consecutive RH elements
99   int kSize = sizeof(kTable)/sizeof(kTable[0]);
100  int  Tdis, RHdis;
101
102 // // Define the variables regarding the filter and the air flow calculation // //
103   float pi        = 3.141592654;        // [-], Mathematical constant pi
104   float rFilter   = 0.04;               // [m], radius of the filter
105   float A         = pi*pow(rFilter, 2); // [m^2], area of the filter
106   float Cd        = 0.02;               // [-], discharge coefficient
107   float R         = 8.314472;           // [J/molK], molar gas constant
108   float Ma        = 28.96546e-3;        // [kg/mol], molar mass of dry air
109   float Mv        = 18.01528e-3;        // [kg/mol], molar mass of water vapor
110   float psv, f, xv, Z, rho, v, Q;       // [:], Intermediate variables, Q is the air flow in
         [L/min]
111
112 // // Define the variables regarding the filter quality // //
113   float PthresholdLow = 70;   // [Pa], EU guideline for maximum allowable pressure drop at
         low air flow
114   float PthresholdHigh = 240; // [Pa], EU guideline for maximum allowable pressure drop at
         low air flow
115   float PdiffMax;             // [Pa], The maximum pressure drop that is measured over time
116   int PdiffTime = 50;         // [ms], The time between consecutive Pdiff measurements
117   int NPdiff = 1200;          // [-], Number of consecutive Pdiff measurements
118   bool indicationSignal;      // Indication signal, high if the pressure drop has become too
         large
119
120 // // Define the variables used for running the programme // //
121   int i, j;                           // Used for running for-loops
122   float I, T, RH, Pabs, Pdiff;        // The values that are measured
123   int LEDR, LEDG, LEDB;               // Used for the indication LED
124   unsigned long timer = 0;            // Used for timing the control loop
125   unsigned long PIDtimer = 0;         // Used for timing the PID loop
126   unsigned long Tremain = 0;          // Used for timing the PID loop
127   unsigned long loopDelay = 1800000;  // The loop will take half an hour
128   unsigned long sleepTime;            // Used for enabling sleep mode
129
130 void setup() {
131   // // Define the input and output pins of the Arduino Pro Mini // //
```

```
132     pinMode(UVsenspin_1, INPUT);
133     pinMode(UVsenspin_2, INPUT);
134     pinMode(UVLEDpin_1, OUTPUT);
135     pinMode(UVLEDpin_2, OUTPUT);
136     pinMode(LEDRpin, OUTPUT);
137     pinMode(LEDGpin, OUTPUT);
138     pinMode(LEDBpin, OUTPUT);
139
140   // // Begin the I2C communication // //
141     Wire.begin();
142
143   // // Begin the serial communication // //
144     Serial.begin(9600);
145     while(!Serial){delay(1);}  // Wait until serial port is opened
146     Serial.println(F("Welcome␣to␣the␣Sensing␣and␣Control␣subsystem␣of␣the␣SPPE!")); Serial.
            println();
147
148   // // Si7021 setup // //
149     //startup and verification commands
150     if (!Si7021.begin()){errorSensorSi7021 = true;}   // True if error occured
151     //debug mode for the sensor turned on, obtain all sorts of information from it:
152     if (Si7021_debugMode == true){
153       Serial.print(F("␣Rev("));  Serial.print(Si7021.getRevision());  Serial.print(F(")"));
154       Serial.print(F("␣Serial␣#")); Serial.print(Si7021.sernum_a, HEX); Serial.println(Si7021
              .sernum_b, HEX);
155       Serial.println();
156     }
157
158   // // LPS33HWTR setup // //
159     // The Library uses the default address 0x5D
160     Serial.println(F("Adafruit␣LPS33HWTR␣Test"));
161     if (!lps33hwtr.begin_I2C()) {Serial.println(F("Could␣not␣find␣LPS33HWTR␣sensor")); while
            (1);}
162     Serial.println(F("Found␣LPS33HWTR␣sensor")); Serial.println();
163     // Set the Output Data Rate (ODR) to one shot mode
164     lps33hwtr.setDataRate(LPS35HW_RATE_ONE_SHOT);
165
166   // // ABPMRRV001PD2A3 setup // //
167     // No further setup required
168
169   // // Initialize the kTable // //
170     // multiply each entry of the kTable by the Kowalsky inactivation constant k0
171     for(i=0;i<kSize;i++){for(j=0;j<kSize;j++){kTable[i][j] = k0*kTable[i][j];}}
172
173   // // PID controller setup // //
174   myPID.SetMode(AUTOMATIC);
175   myPID.SetOutputLimits(PIDoutMin, PIDoutMax);
176
177   // // Initial measurements and filter quality determination // //
178     Serial.println(F("Setup␣completed.␣Intial␣values␣of␣the␣sensors:␣")); Serial.println();
179     T = getT(busSi7021_1);
180     RH = getRH(busSi7021_1);
181     T = getT(busSi7021_2);
182     RH = getRH(busSi7021_2);
183     Pabs = getPabs();
184     Pdiff = getPdiff();
185     Q = airFlow(T, RH, Pabs, PdiffMax);                  // Calculate the air flow in the
            filter
186     indicationSignal = determineIndSignal(Q, Pdiff);     // Determine if the threshold has
            been reached
187     setIndicationSignal(indicationSignal);               // Set the indication signal
188     Serial.println(); Serial.println(F("Enter␣continuous␣operation␣of␣the␣SPPE's␣Sensing␣and␣
            Control␣subsystem."));
189     Serial.println();
190 }
191
192 void loop() {
193   timer = millis();                        // Set the timmer to the current time
194
195   Imin = referenceFunction(busSi7021_1);  // Calculate Imin using the reference function for
          LED array 1
```

```
196   PIDcontrol(UVLEDpin_1, Imin);           // Perform the PID control for LED array 1
197
198   Imin = referenceFunction(busSi7021_2);  // Calculate Imin using the reference function for
          LED array 2
199   PIDcontrol(UVLEDpin_2, Imin);           // Perform the PID control for LED array 2
200
201   FilterQuality();  // Filter quality functionality
202
203   timer = millis()-timer;                 // timer now containts the loop time
204   sleepTime=loopDelay-timer;              // The remaining time
205   // Sleep and wake-up every second, stay awake when the time is almost over
206   while(sleepTime>10000){sleepTime = sleepTime-Watchdog.sleep(1000);}
207 }
208
209
210 // UV control loop functions // //
211   float referenceFunction(int selectSi7021){
212     RH = getRH(selectSi7021);               // Measure RH
213     T = getT(selectSi7021);                 // Measure T
214     RHdis = round(RH/RHinterval);           // Round off to the intervals
215     Tdis = round(T/Tinterval);              // Round off to the intervals
216     if(0<=Tdis<kSize && 0<=RHdis<kSize)
217     {kVal = kTable[Tdis][Tdis];}            // Determine which value for k is used
218     else{kVal = 0.7*k0;}
219     Dmin = -1*log(S0)/kVal;                 // [J/m^2] Minimum dose
220     Imin = Dmin/t0;                         // [W/m^2] Minimum intensity
221     return(Imin);
222   }
223
224   void PIDcontrol(int selectUV, float Imin){
225     PIDref = Imin;
226     Tremain = t0;
227     while(Tremain > t0/100){
228       PIDtimer = millis();                  // The current time
229       PIDin = getI(selectUV);               // Measure the UV intensity
230       myPID.Compute();                      // Calculate PIDout
231       analogWrite(selectUV, PIDout);        // Provide the output to the UVGI subsystem
232       indicationLED(255, 0, 255);           // The RGB LED is set to purple to indicate UV
              radiation to the user
233
234       PIDtimer = millis()-PIDtimer;         // The time the PID control took
235       Tremain = Tremain - PIDtimer;         // The time remaining of t0
236     }
237   }
238
239
240 // // Filter quality functions // //
241   void FilterQuality(){
242     RH = getRH(busSi7021_1);                              // Update the value of RH,
          arbitrary sensor
243     T = getT(busSi7021_1);                                // Update the value of T, arbitrary
           sensor, but same as for RH
244     PdiffMax = peakPdiffDetection();                      // Measure the maximum pressure
          difference
245     Pabs = getPabs();                                     // Measure the absolute pressure
246     Q = airFlow(T, RH, Pabs, PdiffMax);                   // Calculate the air flow in the
          filter
247     indicationSignal = determineIndSignal(Q, PdiffMax);   // Determine if the threshold has
          been reached
248     setIndicationSignal(indicationSignal);                // Set the indication signal
249   }
250
251   float peakPdiffDetection(){
252     Serial.println(); Serial.println(F("Starting␣Pdiff␣measurement␣stream"));
253     PdiffMax = Pdiff;                           // Intial value of the maximum pressure drop
254     for(i=0;i<1200;i++){                        // 1200*0.050 = 1 minute (exc. communication
          time)
255       Pdiff = getPdiff();
256       if(Pdiff > PdiffMax){PdiffMax = Pdiff;}  // Update the highest pressure drop
257       delay(37); // As 13 ms of sensor measurement time is added to this
258     }
```

```
259     return(PdiffMax);
260   }
261
262   float airFlow(float T, float RH, float Pabs, float PdiffMax){
263     psv = exp(1.2378847*pow(10,-5)*pow(T, 2) - 1.9121316*pow(10,-2)*T + 33.93711047 -
            6.3431645*pow(10,3) /T); // [Pa], vapour pressure at saturation
264     f = 1.00062 + 3.14*pow(10,-8)*Pabs + 5.6*pow(10,-7)*pow((T-273.15),2); // [-],
            enhancement factor
265     xv = (RH/100)*f*psv/Pabs; // [-], mole fraction of water vapour
266     Z = 1 - (Pabs/T)*(1.58123*pow(10,-6) - 2.9331*pow(10,-8)*(T-273.15) + 1.1043*pow(10,-10)*
            pow((T-273.15),2) + (5.707*pow(10,-6)
267         - 2.051*pow(10,-8)*(T-273.15))*xv + (1.9898*pow(10,-4) - 2.376*pow(10,-6)*(T-273.15))
                *pow(xv,2))
268         + pow((Pabs/T),2)*(1.83e-11 - 0.7564e-8*pow(xv,2)); // [-], compressibility factor
269     rho = (Pabs*Ma)/(Z*R*T) * (1-xv*(1-Mv/Ma)); // [kg/m^3], Density of the air in the SPPE
270
271     v = Cd*sqrt(2*Pdiff)/sqrt(rho); // [m/s], Velocity of air in the SPPE
272     Q = A*v;             // [m^3/s], Air flow in the SPPE
273     Q = Q*1000*60;       // [L/min], Air flow in the SPPE
274     return(Q);
275   }
276
277   bool determineIndSignal(float Q, float PdiffMax){
278     indicationSignal = 0;
279     if(Q <= 95){
280       if(PdiffMax >= PthresholdLow){
281         RH = getRH(busSi7021_1);                // Update the value of RH, arbitrary sensor
282         T = getT(busSi7021_1);                  // Update the value of T, arbitrary sensor,
                but same as for RH
283         PdiffMax = peakPdiffDetection();        // Measure the maximum pressure difference
284         Pabs = getPabs();                       // Measure the absolute pressure
285         Q = airFlow(T, RH, Pabs, PdiffMax);     // Calculate the air flow in the filter
286         if(Q <= 95){if(PdiffMax >= PthresholdLow){indicationSignal = 1;}}
287         else{if(PdiffMax >= PthresholdHigh){indicationSignal = 1;}}
288       }
289     }
290     else{
291       if(PdiffMax >= PthresholdHigh){
292         RH = getRH(busSi7021_1);                // Update the value of RH, arbitrary sensor
293         T = getT(busSi7021_1);                  // Update the value of T, arbitrary sensor,
                but same as for RH
294         PdiffMax = peakPdiffDetection();        // Measure the maximum pressure difference
295         Pabs = getPabs();                       // Measure the absolute pressure
296         Q = airFlow(T, RH, Pabs, PdiffMax);     // Calculate the air flow in the filter
297         if(Q <= 95){if(PdiffMax >= PthresholdLow){indicationSignal = 1;}}
298         else{if(PdiffMax >= PthresholdHigh){indicationSignal = 1;}}
299       }
300     }
301     return(indicationSignal);
302   }
303
304   void setIndicationSignal(bool indicationSignal){
305     if(indicationSignal == 1){indicationLED(0, 0, 255);} // Set the indication LED blue,
            indicates a clogged filter
306     else{indicationLED(0, 0, 0);}                       // Set the indication LED low
307   }
308
309   void indicationLED(int LEDR, int LEDG, int LEDB){
310     analogWrite(LEDRpin, LEDR);             // Radiate red light
311     analogWrite(LEDGpin, LEDG);             // Radiate green light
312     analogWrite(LEDBpin, LEDB);             // Radiate blue light
313   }
314
315
316 // // Sensor measurement functions // //
317   void TCA9548A(uint8_t busSelect){
318     Wire.beginTransmission(I2CMUXaddress);  // Select the I2C device
319     Wire.write(1 << busSelect);             // Send a byte to indicate which bus (i.e. which
            sensor) is used
320     Wire.endTransmission();                 // End the communication, dit zou hier al moeten
            volgens de tutorial
```

```
321   }
322
323   float getT(int selectSi7021){
324     TCA9548A(selectSi7021);                 // Select the correct I2C bus on the I2C MUX
325     T = Si7021.readTemperature();           // [degree C], Returns the T
326     Serial.print(F("Temperature␣of␣sensor␣")); Serial.print(selectSi7021); Serial.print(F("␣
          is:␣")); Serial.print(T); Serial.println(F("␣degree␣C"));
327     return(T);
328   }
329
330   float getRH(int selectSi7021){
331     TCA9548A(selectSi7021);                 // Select the correct I2C bus on the I2C MUX
332     RH = Si7021.readHumidity();             // [%], Returns the RH
333     Serial.print(F("Relative␣humidity␣of␣sensor␣")); Serial.print(selectSi7021); Serial.print
          (F("␣is:␣")); Serial.print(RH); Serial.println(F("␣%"));
334     return(RH);
335   }
336
337   float getPabs(){
338     lps33hwtr.takeMeasurement();            // required to make the sensor awakes from sleep
          mode
339     Pabs = 100*lps33hwtr.readPressure();    // [Pa], Returns the absolute pressure
340     Serial.print(F("Absolute␣pressure:␣")); Serial.print(Pabs); Serial.println(F("␣Pa"));
341     return(Pabs);
342   }
343
344   float getPdiff(){
345     ABP.update();
346     Pdiff = 6894.75729*ABP.pressure();      // [Pa], Returns the differential pressure
347     Serial.print(F("Differential␣pressure:␣")); Serial.print(Pdiff); Serial.println(F("␣Pa"))
          ;
348     return(Pdiff);
349   }
350
351   float getI(int selectUVsens){
352     I = analogRead(selectUVsens);           // Returns the UV intensity as a number in the
          range of [0, 1023]
353     I = map(I, 0, 1023, 0, 10.45);          // [W/m^2], the measured UV intensity
354     Serial.print(F("UV␣intensity:␣")); Serial.print(I); Serial.println(F("␣W/m^2"));
355     return(I);
356   }
357
358
359 // // End of file // //
360
361 /*
362   Copyright text of the Adafruit libraries.
363   First, for the Adafruit Sleepydog library:
364
365   The MIT License (MIT)
366
367   Copyright (c) 2015 Adafruit Industries
368
369   Permission is hereby granted, free of charge, to any person obtaining a copy
370   of this software and associated documentation files (the "Software"), to deal
371   in the Software without restriction, including without limitation the rights
372   to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
373   copies of the Software, and to permit persons to whom the Software is
374   furnished to do so, subject to the following conditions:
375
376   The above copyright notice and this permission notice shall be included in all
377   copies or substantial portions of the Software.
378
379   THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
380   IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
381   FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
382   AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
383   LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
384   OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
385   SOFTWARE.
386
```

```
387
388   Second, for the Adafruit LPS35HW library:
389
390   Software License Agreement (BSD License)
391
392   Copyright (c) 2012, Adafruit Industries
393   All rights reserved.
394
395   Redistribution and use in source and binary forms, with or without
396   modification, are permitted provided that the following conditions are met:
397   1. Redistributions of source code must retain the above copyright
398   notice, this list of conditions and the following disclaimer.
399   2. Redistributions in binary form must reproduce the above copyright
400   notice, this list of conditions and the following disclaimer in the
401   documentation and/or other materials provided with the distribution.
402   3. Neither the name of the copyright holders nor the
403   names of its contributors may be used to endorse or promote products
404   derived from this software without specific prior written permission.
405
406   THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS ''AS IS'' AND ANY
407   EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
408   WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
409   DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER BE LIABLE FOR ANY
410   DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
411   (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
412   LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
413   ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
414   (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
415   SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
416 */
```