Measuring thermal conductivity of textiles for time of death estimation

By

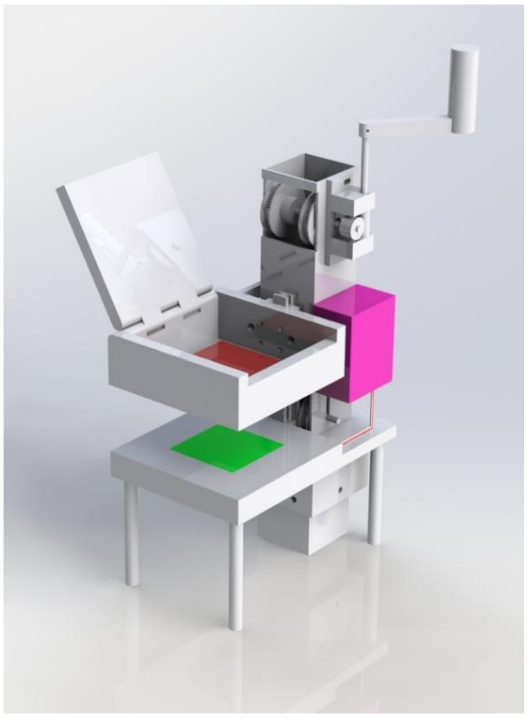
Ivo Sebastiaan Best (4484509)
Nicolaas Sebastian Noort (4087445)
Sebastiaan Thomas Njio (4517393)
Jason Morgan van Schijndel (4490339)
Fabian Matthias Verhage (4591801)

CONFIDENTIAL

Supervisor: Dr.ir. A.Loeve
Delft University of Technology

THERMINUS

Financed by: Innovatieteam Ministerie van Justitie en Veiligheid



Abstract 3
I. Introduction 3
II. Design process 3
A. Design requirements 3
B. Functional design and justifications 3
1) Conductivity measurement method 3
2) Insulation 4
3) Temperature Difference 4
4) Temperature Control 6
C. Prototype design 6
1) Mechanical design 6
a - Defining partial solutions 6
b - Material Selection 7
2) Electrical design 7
a - Heat Pad 7
b - Peltier Element 8
c - Temperature Sensors 8
d - Arduino and temperature control 8
e - MOSFET For PWM Control 8
III. Concept Validation 8
IV. Results 9
V. Discussion 9
A. Limitations and recommendations for future research 9
VI. Conclusion 10
Acknowledgements 10
Sources 10
Appendix I: Measuring principles 1
Guarded hot plate method 1
Hot wire method 1
Modified Transient Plane Source method (MTPS) 2
Appendix 2: Phoebe experiment 3
Appendix 3: Morphological chart 5
Appendix 4: Solidworks assembly drawing 8
Appendix 5: Insulation 9
Insulation 9
Shielding 9
Assumptions 9
Values 10
Most Critical Case 10

Important remarks 12
Outputs from MATLAB script 12
Time to Steady State 12
Appendix 6: Heat pad 26
Appendix 7: Electrical circuit 30
Appendix 8: Arduino code 34
Arduino Code 35
Appendix 9: Thermal conductivity measurement uncertainty 40
Defining Aspired Precision 40
Parameter Remarks 40
General Formula Error Propagation 40
Formulae ΔT Error Propagation 41
Formulae $dtextile$ Error Propagation 41
Formulae Q Error Propagation 41
Resulting Uncertainty Formula 42
Two Evaluated Cases 42
Uncertainty Values 42
Resulting Error Estimate 43
Lineair Interpolation Heat Loss Model Justification 43
Appendix 10: Thermistor uncertainty 46
Thermistor Sensor Setup 46
Parameters as Used in Error Propagation Appendix 47
MATLAB Script Outputs 47
Appendix 11: Linear potentiometer uncertainty 51
Appendix 12: Heat loss insulation uncertainty 55
Heat Loss Interpolation 55
Parameters as Used in Error Propagation Appendix 55
Outputs by MATLAB script 56
MATLAB Code 56
Appendix 13: General project planning 58
Appendix 14: Clothing thickness dataset 59

Abstract

This project was conducted as part of the research project Therminus, a collaboration between the Netherlands Forensic Institution (NFI), the Amsterdam Academic Medical Centre (AMC) and the TU Delft, which was initiated with the goal to increase the accuracy of time of death estimations. The objective of this project was to design and produce a prototype which can accurately measure the thermal conductivity and thickness of a sample of the clothing worn by the deceased. A design process was undergone which resulted in the construction of an initial prototype. At the TU Delft no suitable conductivity measurement device was available for checking the experimental measurements of the prototype. Thus, validation of the prototype was done by measuring the conductivity of materials with known thermal properties. Initial testing showed that the prototype did not accurately measure thermal conductivity. This was deemed to be due to the fact that actual heat loss was higher than calculated. After adjusting the heat loss and testing paper and cotton samples conductivity values comparable to tabulated values were found by the device. In order to build on the outcomes of this project, future research could be done focusing on investigating to what extent a relationship exists between the level of compression of a material and the value of thermal conductivity which is found.

I. Introduction

In 2017 the lives of 158 people were taken by homicide in the Netherlands [1]. The resulting investigations were not always conclusive. In fact, in 2017 less than 26% of registered crimes in the Netherlands were resolved [2]. In order to resolve more murders the exact Time of Death (ToD) of the victim should be determined. This would aid the process of identifying and eliminating suspects. Currently the ToD estimation is done using the Henssge model. This method uses the ambient body temperature, rectal body temperature and bodyweight of the deceased [3]. However, the model relies on various assumptions and there are many conditions under which it can't be applied. This makes the model user-dependent and it has a high uncertainty. A more intricate ToD estimation, called Phoebe [3], is created to replace the outdated Henssge model. In Project Therminus methods of measuring thermal properties of clothes and surfaces are being developed in order to use the properties as input in the Phoebe model. Within this project the focus lies on measuring the thermal conductivity of textiles. Clothes act as insulators to keep us warm and protected from the elements, but the conductivity of clothing varies across different kinds of fabrics. The aim of the project is to design and produce a prototype which can accurately measure the thermal conductivity and thickness of a sample of the clothing worn by the deceased.

II. Design process

A. Design requirements

As the device must be suitable for use at crime scenes, it must conform to several requirements determined in previous research on the project. First and foremost, the device must be able to measure the thermal conductivity of any clothing sample with an error small enough that it leads to an error in the ToD estimation of less than 3 percent [3, 8]. This 3 percent error in the ToD estimation translates to an 8 percent maximum error in the thermal conductivity and was found by testing the Phoebe model at the thermal conductivity level which has the lowest acceptable absolute error as stated Appendix in 2 Phoebe experiment. The clothing sample can be cut, but damage to materials should be minimal [3]. Additionally, the device must conduct its conductivity measurement within a timeframe of less than 30 minutes, since forensicists have limited time at crime scenes [9, 3, 8]. On top of that, no cross-contamination may take place, which means the device should be either cleanable or disposable [9]. Cleaning will be done with alcohol, chlorine and UV-light. Finally, the device should be low cost and portable [9].

These requirements form the basis for the design of the device. In the next paragraph design choices are outlined. Each design choice is justified by the fact that it contributes to the fulfilment of least one requirement.

B. Functional design and justifications

1) Conductivity measurement method

Several methods of measuring the thermal conductivity of a material exist. The choice of working principle for the design process undergone in this project was made by evaluating working principles discussed in previous research [3]. Particular consideration was given to 3 concepts, which are briefly summarized in Appendix 1 Measuring principles. After analysing the three methods, a steady state method with 1-directional heat flow as visualised in figure 1 was deemed most suitable for this project. This method involves placing the sample material in between two aluminium plates, one of which is heated by a heat pad and one of which is cooled by a Peltier element. The heat flows through the sample from the hot body to the cold body, which are both kept at a constant temperature. This setup is surrounded by a strong insulator, so as to minimize loss of heat to the environment. The amount of heat added can be chosen, and given that the temperatures of the cold and hot bodies are held constant, such that thermal conductivity k can be determined using Fourier's law of heat conduction (1) [7]:

$$Q = \frac{A}{d} k \, \Delta T \qquad (1)$$

In which Q is the amount of heat that is conducted through the sample, A is the area of the hot and cold body, d is the thickness of the sample and ΔT is the temperature difference.

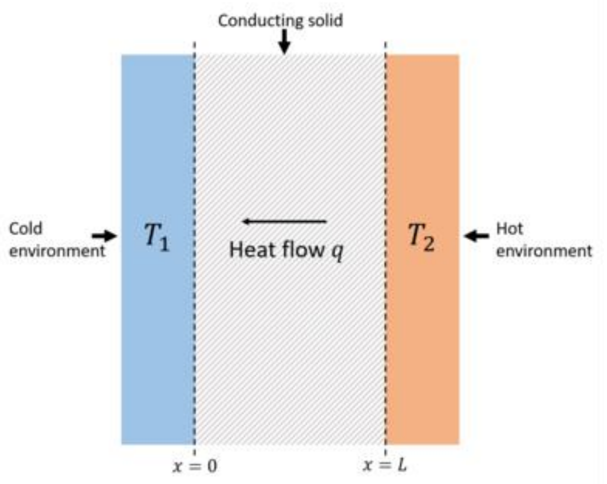


Figure 1: Schematic representation of a steady state method with 1-directional heat flow [6]

Because the aforementioned measurement method relies on a one-directional heat flow, it is crucial for the accuracy of results that most of the heat flows in the predetermined direction, and that the fraction of heat lost can be accurately estimated. The following paragraph describes how this has been ensured and why the design choice for insulation was made.

2) Insulation

In order to achieve reliable estimations of the thermal conductivity, measurements must be made in a controlled environment. Since crimes can take place in varying settings, the environment contains unknown factors which can distort the measurements. Examples of such unknowns are airflow due to the wind, which leads to thermal convection, and an unknown radiative heat transfer with the environment. To eliminate these unknowns the device must be insulated. An important result of this is a reduction of heat transfer to the surrounding environment and less uncertainty regarding the flow of heat. As insulation material Polyisocyanurate (PIR) was chosen, due to the fact that it is one of the strongest insulators available, having a thermal conductivity of 0.022 W/(mK). In choosing the thickness of the insulation a trade-off had to be made. Given that the device must be portable, a compact design is required, meaning that the insulation can not be too thick. A compact design also reduces the cost of materials. On the other hand, the insulation has to be thick enough to reduce the amount of heat lost to the environment to an acceptable level.

Appendix 5 Insulation, gives a proper in depth explanation on which insulation thickness was chosen. The result is an insulation thickness of 30mm. This gave a good trade-off between size, insulation effectiveness and time it takes to reach steady state operation. Figure 2 shows that, after 30mm of insulation thickness the heat loss through the insulation flattens this means a larger thickness only provides a minor increase in insulation effectiveness.

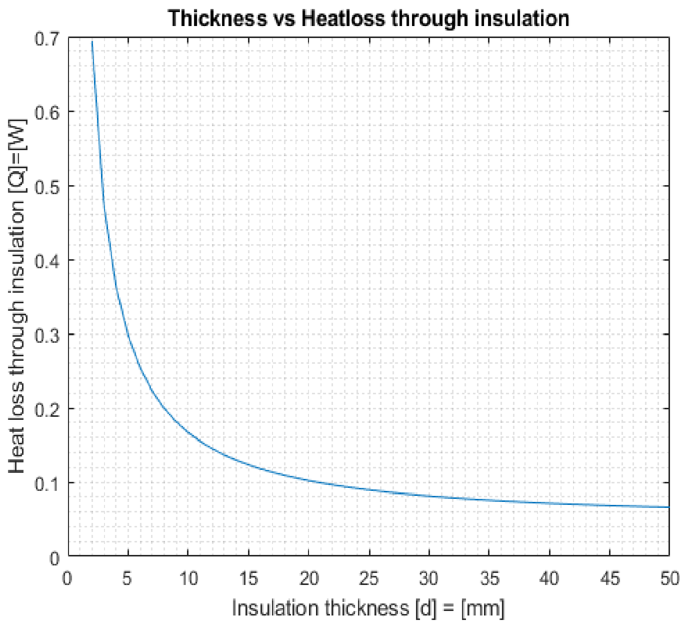


Figure 2: Power loss through the insulation in comparison to the insulation thickness.

3) Temperature Difference

A temperature difference between the insulation surface and the environment temperature is to be expected. Appendix 5 Insulation describes the method used for evaluating the exact heat flow through the proposed insulation. The insulation is to separate the environment from the hot body which will be kept at a 10 degree temperature difference with respect to the environment. The radiative heat transfer is hard to account for in an unknown environment, thus in order to increase the accuracy of the Therminus device, shielding is proposed. Shielding the device has two beneficial effects. The first beneficial effect is that unknown radiation sources, from the environment, will not distort measurements. The second benefit is that a controlled environment, around the insulation, is created. The controlled environment allows for the description of both radiative and convective heat transfer. The thermal conductivity of a specimen is determined using equation (2).

(2) $k_{textile} = d_{textile} * (\frac{A*\Delta T}{Q} - 2 * \frac{d_{element}}{k_{element}})^{(-1)}$

Q as present in equation (2), is the amount of heat that is conducted through the specimen. The value for Q is determined by use of equation (3). Thus requiring description for both the heat loss through insulation $Q_{insulation}$, and $Q_{supplied}$.

(3) $Q = Q_{supplied} - Q_{insulation}$

Minimizing $Q_{insulation}$ with respect to the Q necessary for measurements, also minimizes any errors caused by mistakes made in the $Q_{insulation}$ estimation.

The surface temperature of the insulation will be logically low (as compared to the environment) since the amount of heat that needs to be dissipated is dispersed over a large surface area. Figure (3) depicts the expected insulation surface temperature for the range of expected environment temperatures. Please note that the surface temperature, as depicted in figure 3, seems to be related in a linearly inversely proportional way. Also note that figure 3 is retrieved from Appendix 5 Insulation. Figure 4 as retrieved from Appendix 13 Heat loss insulation uncertainty, depicts the absolute error made when assuming linear interpolation. The significance of the aforementioned assumption is defined in Appendix 10 Thermal conductivity measurement uncertainty. The error made by assuming linear interpolation as a valid method for estimating the heat loss through insulation is small and constitutes a small amount (~6%) of the total error. Please note that this value is subject to change when using a different voltage measurement logic board. Details of which are to be found in Appendix 10 Thermal conductivity measurement uncertainty.

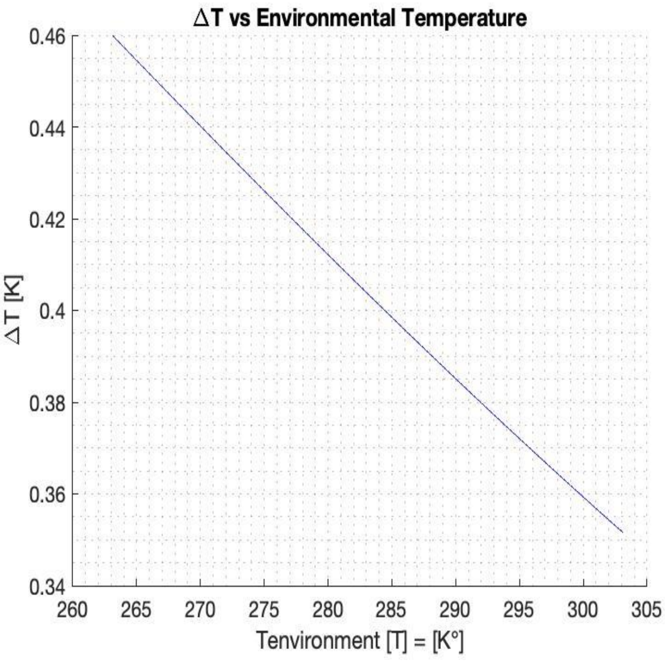


Figure 3: Expected temperature difference between insulation surface and environment temperature

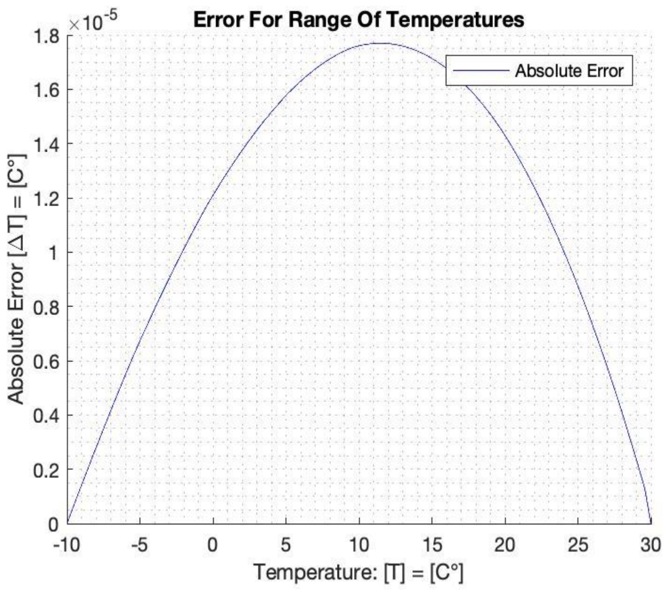


Figure 4: Absolute Error when assuming linear interpolation insulation heat loss model

4) Temperature Control

To maintain the aforementioned 10 degrees temperature difference, the heat pad will have to operate at a range of 0.15W to 40W see Appendix 6 Heat pad. Also, the peltier element will have to remove heat from the system in order to keep the cold body at the environment temperature. The current within the electronic circuit cannot be easily manipulated. Therefore, both heating and cooling elements are controlled using the onboard PWM controller on an Arduino. An Arduino was chosen to control this signal due to having experience with using one in earlier projects. The maximum current draw of the heat pad and peltier element is 4000 mA. This high current poses a challenge, as regular Bipolar Junction transistors (BJT) cannot efficaciously switch such high currents without burning out. Therefore BJT transistors are omitted from the design considerations. The solution for the proposed problem is stated in the electrical circuit design.

C. Prototype design

In this section, both the mechanical and electrical design of the prototype are discussed. The mechanical design subsection describes partial solutions to the stated design problems, as well as displaying the SolidWorks model and justifying the choice of materials. The electrical design subsection describes a number of important electrical components and their accuracy. Additionally, control mechanisms and circuitry considerations are mentioned.

1) Mechanical design

a - Defining partial solutions

At the start of the project a morphological chart was created, Appendix 3 Morphological chart, outlining problems and potential solutions discussed in previous research on the Therminus project [1,8]. The chart can be found in Appendix 4 Solidworks assembly drawing.

The main problems identified were the following:
-How will the steady-state system be kept at a stable temperature difference? (1)
-How will the system be insulated from the environment to reduce heat leaks? (2)
-How will the device bring the hot and cold parts together as to measure the sample? (3)

Several solutions were found for these questions. The final design can be found in figure 5.
The solution for (1) is reliant on two subsystems. The hot body is heated by a heat pad to the desired constant temperature. The cold body is held at environment temperature with a peltier element{13}, heat sink{20} and fan combination.

The insulation (2) for the hot body system has to ensure that the heat loss (to the environment) is minimal in order to have an accurate estimation of how much heat is conducted by the sample. As solution the hot body is contained in a hollow box{4} filled with insulation material made to size. The respective box has square openings to allow for assembly. Temperature sensors

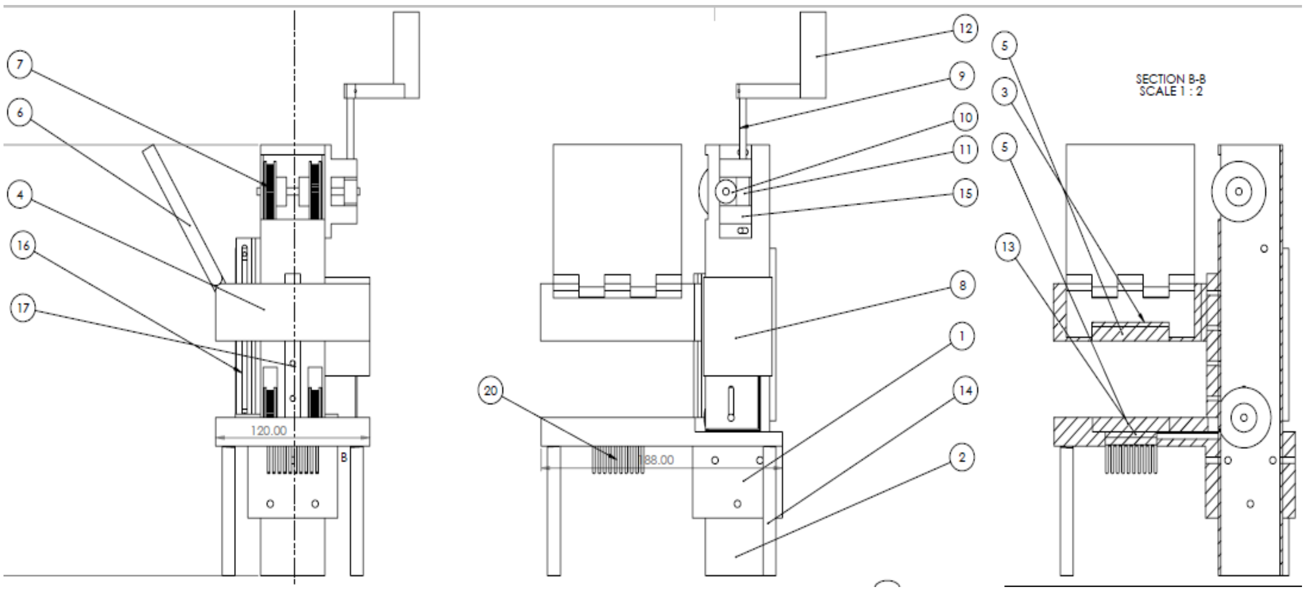


Figure 5: Part of the drawing of the assembly from the SolidWorks model with numbers referencing to its respective parts. The rightmost drawing is a view through the centre of the model to outline internal parts

are placed on the surfaces of both the hot and cold body{5}, whilst opposing the sample of which the thermal conductivity is to be measured. The placement of the temperature sensors results in more cumbersome mathematical formulae, but is offset with the increased usability and cleanability of the device. The used formulae with the respective uncertainty analysis can be referenced in detail by consulting Appendix 10 Thermal conductivity measurement uncertainty.

For the last problem (3) a linear rail guide{17} is used. The bottom part{1} is the cold body system and will be kept stationary and the hot body will be able to move up and down. The box of the bottom part will be fitted with an extruded piece, in which a vertical square tube{2} is mounted. On the vertical square tube a linear rail guide is mounted, on which the hot body box is assembled. The guide rail will add additional rigidity to the system. The propulsion system consists of a worm drive and belts{10,11}. The worm drive is self-locking, making it possible to accurately set elevation levels on the system. A linear potentiometer{16} is assembled on the square tube and attached to the hot-body box in order to measure the distance between the hot-body and cold-body boxes. The addition of a linear potentiometer is justified by the requirement of Phoebe (imposed on the user) to input a thickness of the analysed specimen.

b - Material Selection

Figure 5 displays drawings of the prototype, as modelled in SolidWorks. Gross dimensions and complete views can be found in Appendix 4 Solidworks assembly drawing. The square tube {2} is made of aluminium, because it is rigid and lightweight. The upper box {4} and lower structure {1} are 3D printed out of PLA. Justification for PLA follows from four properties being that PLA is lightweight, a good insulator (k=0.13 W/mK), inexpensive and can be made to the desired dimensions. The supporting rods {14} and drive rods {9} are made of steel instead of aluminium, as these parts are considered critical mechanical parts of the device. The conductive material {5}, previously referred to as hot and cold body, aluminium Alloy 5754 (k=147 W/mK) is chosen. This specific alloy has a high thermal conductivity, is extremely resistant to corrosion and resistant to the mentioned cleaning methods. Both PLA and aluminium meet the previously defined design requirements imposed on the materials.

2) Electrical design

a - Heat Pad

A key requirement, the device has to fulfil, is that it must be capable of measuring a range of thermal conductivity values. The chosen conductivity measurement method, as described on page four, relies on a heat pad to provide the heat which is to be conducted through the sample. This is considered adequate for the Therminus device. The heat pad is able to create the desired 10 degree temperature difference. The thermal aspect of the measurement depends on a few design parameters, namely: surface area, thickness, thermal conductivity and temperature difference. As stated before, the relationship between specimen, heatflow and geometric properties are given by fourier's law of steady state thermal conductivity which states that

(1) $$Q = \frac{A}{L} k \, \Delta T$$

The values for thermal conductivity that the Therminus device should be able to measure range from 0.057 on the low end, and 0.52 on the high end of the spectrum [14]. Data based on a sample size of 25 T-shirts, see Appendix 14 Clothing thickness dataset suggests that 99.73% of T-shirts is expected to be at least 0.44 mm thick. Thus suggesting a minimum thermal resistivity of $[R_{th}]$ =8.46*10^{-4} [m^2K/W]. Once more assuming clothing thickness is normally distributed, in 99.73% of cases, coats will be thinner than 28mm. The upper bound for thermal resistivity thus becomes $[R_{th}]$ =4.80*10^{-02} [m^2K/W].

From Fourier's law of thermal conductivity a design parameter is derived (4)

(4) $$\frac{Q}{A} = \frac{\Delta T}{R_{th}}$$

One can deduct that the head pad is required to have a minimum index (Q/A) > 11.8*10^3. The chosen heat pad has a power of 48W heat pad and a surface area of 0.036 m^2. The heat pad has a performance index (Q/A) of 13*10^3 and will therefore be able to supply sufficient heat for at least 99.73% of T-shirts. Please consult Appendix 6 Heat pad for the used formulae and the used MATLAB script.

b - Peltier Element

A peltier element is used to remove heat that is conducted by the sample, in order to maintain the desired temperature difference. The element is shaped like a square and transfers heat from one surface to the other. Peltier is fixed to the cold aluminium body and is PID controlled, see Appendix 8 Arduino code for the PID control code. Since the peltier element is merely used to remove heat from the system (to maintain steady state), said element does not detract from the accuracy of the device. The former claim is used as justification to omit said peltier from the uncertainty analysis as carried out in Appendix 10 Thermistor uncertainty. Uncertainties related to the cold body are inaccuracies which are primarily attributable to the electrical temperature measurement setup.

c - Temperature Sensors

Three temperature sensors are used in the prototype. The environment temperature sensor is a PTC sensor, meaning its resistance increases with temperature, with a base resistance of 2 kΩ at 25 degrees Celsius. The other two sensors are thin film sensors, designed specifically to measure surface temperatures. Their thin construction enables fast temperature response, allowing for quick and accurate surface temperature measurements. These sensors are placed against the hot and cold bodies. For these sensors a higher base resistance, 5kΩ at 25 degrees Celsius, was chosen. All three sensors can viably be used in temperatures ranging from -50 to 150 degrees Celsius and have a prespecified accuracy of 1%. The process of calibration lowers the maximum temperature uncertainty from 6.758% to 1.278%, see Appendix 10 Thermistor uncertainty. For the calibration process see Appendix 7 Electrical circuit.

d - Arduino and temperature control

The prototype uses an arduino UNO to power and/or control the electrical components used. See Appendix 7 Electrical circuit for the electrical circuit and how everything is connected to each other. Temperature control is handled by means of a PID controller (see Appendix 8 Arduino code) implemented in the arduino software uploaded onto the arduino. The choice for a software based control system comes mainly from usability and flexibility point of view. Two PID controllers are used in conjunction to control the duty cycles of both the heat pad and the peltier element respectively. Both duty cycles are controlled through PWM. PWM allows the heat pad and peltier to function at the desired voltage and current whilst also enabling control. An important remark is that the Arduino Uno does possess a 16-bit timer which can be enabled through low level software modifications. Whilst theoretically possible, one might refrain from doing so since this can have negative consequences for the usage of standard Arduino libraries.

e - MOSFET For PWM Control

The heating and cooling elements require currents (4000 mA) which Arduino (max 40 mA) is incapable of supplying. For this reason the required current is provided by external power supplies, thereby not passing through the delicate Arduino circuitry. Standard issue semiconductor transistors cannot switch such high currents effectively, which is why the use of MOSFETs is justified. A suitable logic-level MOSFET is the IRLZ44N by International Rectifier. The IRLZ44N can switch the required current without introducing noticeable transient behaviour since its transient spans ~100ns.

In the worst imaginable case the resistance of the MOSFET will amount to 32 μΩ causing a voltage drop of ~12.8 μVolt. The voltage drop (worst case: $<1.06*10^{-04}$ % of smallest voltage drop) incurred by the MOSFET, is small and thus will not be considered in further analysis (error propagation and device performance). The MOSFET's are also suitable for use in a PWM controlled circuit because their transient behaviour is excellent when considering the limitations imposed by the arduino PWM controller. The PWM controller on the arduino board has a minimum duty cycle of 3.8 μs whilst the rise and settle time of the MOSFET's, combined, do not exceed 100 ns. Since the MOSFET's are switching currents far below their maximum currents, the maximum operating temperature never approaches the T-junction, therefore no heat sinks are required.

III. Concept Validation

The prototype has to be tested in order to prove that the concept of choice results in a working device. Looking at the design requirements mentioned in the design process section, it can be said that the accuracy of the device's conductivity measurement should be evaluated in order to conclude whether or not the device functions at an acceptable level. One way to test this is to have the conductivity of test samples measured by a validated measurement device in a university laboratory or through a third party. However, at the TU Delft no suitable conductivity measurement device is available for checking the measurements of the prototype. After consulting with a professor of thermodynamics at the TU Delft, the conclusion was drawn that a second validated measurement device would have to be designed to validate the prototype. This is not feasible within this project. Therefore, validation of the prototype must be done by measuring the conductivity of materials with known thermal properties. Since the apparatus is designed to measure textile samples, which have a low conductivity [3], samples with known values for conductivity rather than table values are required to assess the accuracy of the device. After searching for suitable materials, the construction grade isolation material expanded polystyrene (EPS) with a conductivity of 0.04W/mK was found. This material has a consistent thermal conductivity value and can easily be manipulated to the desired shape.

The difference between the theoretical and experimental value is the error, which has to be reduced to a maximum error of 8%.

IV. Results

During preliminary testing the device did not accurately measure the thermal conductivity values of the verification sample. For the EPS sample mentioned above, a value of 0.04 W/mK was meant to be found, but the value found by the device was 0.4 W/mK. After adjusting the potentiometer settings and properly pressing the hot body onto the material a value of 0.17 W/mK was found.

Although this value was closer to the real conductivity of the EPS sample, the measurement was still inaccurate. To correct for this the absolute heat loss through the insulation was adjusted in the model and eventually the right value of 0.04 W/mK was found for EPS.

To check whether the device would now also provide accurate measurements for other materials, a test was done with paper, which was found to have a thermal conductivity value of 0.05W/mK [12]. Additionally, a test was done with a t-shirt of 100% cotton for which the tabulated value was 0.029W/mK [12]. With the adjusted heat loss values, conductivity values were found by the device which are comparable to the values tabulated in the Engineering Toolbox [12].

V. Discussion

The wrong conductivity values initially given by the device (see Results) where fixed by changing the expected power loss through the insulation. The expected theoretical value is 0.078W, Appendix 5 Insulation and was consequently changed to 0.5W. This is believed to be a justified change because the proposed insulation differs from the realized insulation. The realized insulation differs from the aspired insulation in a few fundamental ways. First of all the real insulation does not fully make contact with the warm body, consequently creating a layer of air with varying thickness between them. The outer layer of the insulation box is made from PLA which has lower thermal conductivity (0.13 W/mK) than the polyisocyanurate (0.022 W/mK). Between this outer layer of PLA and the polyisocyanurate another layer of air can be found, again of varying thickness. It is imperative to note that depending on the air pocket size, internal convection can occur. Internal convection will further degrade the insulation effectivity. Secondly, the device does not yet have the proposed shield, which means forced convection and unknown radiative heat transfer might occur. Lastly, two metal bolts are inserted into the insulation which are connected to two aluminium bodies for belt connection. Those two aluminium bodies are on the outside of the insulation, practically acting as a heatsink.

A. Limitations and recommendations for future research

One limiting factor in the accuracy of the prototype is the fact that the Arduino UNO is practically limited to 8 bit PWM output which means that it can only control duty cycles of the heat pad and peltier element in 256 steps. In part due to the discrete nature of voltage measurement chips, the worst case thermal conductivity error expected to be 22% (see Appendix 9 Thermal conductivity measurement uncertainty). This error could be reduced by switching to a development board with a 12 bit read functionality. An example of such a development board is the Arduino DUE. However, this board implements the SAM3X chip which is limited to inputs of 3.3V maximum. Consequently, switching from the UNO to the DUE was not feasible do to the electrical circuit, as it would entail having to redesign the electrical circuit, recalibrate the PID controller, potentiometer and temperature sensors. It is recommended to use the Arduino DUE in future Therminus projects, Appendix

9 Thermal conductivity measurement uncertainty contains the justification for the recommendation.

Another issue that occurred was heating of the cables. The resistance of the arduino cables combined with the current passing through the cables, caused the cables to heat up over time. To avoid melting cables, the prototype was not used at maximum power. This meant both an increase in the time until achieving steady state and that the device could not measure the full range of textile samples. The problem can be solved by using cables with a larger cross section to reduce their resistance.

A number of recommendations can be given for the mechanical improvement of this prototype. The linear potentiometer should be attached closer to the upper box to improve accuracy due to the reduced bending moment. The linear rail should be of the ball bearing variety instead of the plastic bearing since the earlier has higher stiffness.

A potential improvement for a second version of the prototype is a more simple transmission system with the use of a spindle attached to the moving box, powered by the upper handle. Such a spindle would reduce the amount of moving parts necessary in the device. Additionally, a containment box should be implemented in the second version. This box surrounds the heat-pad and heatsink systems and allows for a controlled environment in order to minimize external effects on the measuring process, like convection and radiation. For the first iteration this box was not made, because validation of the design occurs indoors in a stable environment.

Another limitation faced in this project is the fact that compression of the fabric might affect the readings of the thermal conductivity of the fabric. This was thought to be the case since air, which has its own thermal conductivity, is pressed out of the material. To investigate the effect fabric compression has on the measurements, an experimental study is required. Future research could therefore be aimed at finding the extent to which a relationship exists between the level of compression of a material and the value of thermal conductivity which is found.

VI. Conclusion

This project was set up to further the progress of the Therminus research project which aims to increase the accuracy of ToD estimations. One objective of the Therminus project was to create a device capable of accurately estimating the thermal conductivity of clothing of the deceased. In this project such a device was designed and a prototype was built. Preliminary testing of the prototype did not yield the expected results. This was in part due to the lack of a validated conductivity measurement device to verify results. Nevertheless, promising steps were made towards the goal of the Therminus project. Important lessons were learnt over the course of the project, resulting in a number of recommendations for mechanical and electrical improvements in future prototypes. In particular, the expected heat loss through the insulation should be more accurately calculable. To this end, the prototype's insulation should more closely resemble the insulation theorised in the design. Additionally, the use of a development board with a 12-bit read functionality would help in attaining the desired accuracy. By generating these new insights this project has furthered the Therminus research project, providing a foundation for future prototyping and testing. Through further research, design and prototyping the target of developing a portable device that can estimate the thermal conductivity of a given textile with a margin of 8% is achievable.

Acknowledgements

A special thanks to:

Dr. Ir. A.J. Loeve	TU Delft
Jos van Driel	TU Delft

Sources

[1] Centraal Bureau voor de Statistiek (2018). Overledenen; moord en doodslag; pleeglocatie Nederland. Retrieved 5 June 2019 from https://opendata.cbs.nl/statline/#/CBS/nl/dataset/81453NED/table?ts=1559398808493
[2] Centraal Bureau voor de Statistiek (2019). Geregistreerde criminaliteit; soort misdrijf, regio. Retrieved 5 June 2019 from https://statline.cbs.nl/Statweb/publication/?DM=SLNL&PA=83648NED&D1=0,3-4&D2=0&D3=0&D4=a&HDR=T&STB=G2,G1,G3&VW=T
[3] Dessing, O. (2018). Surface testing methods to define thermal properties of surfaces at crime scenes. TU Delft, Delft
[4] Faghani, F. (2010) Thermal conductivity measurement of pedot : pss by 3-omega technique. Master's thesis, Linkping University
[5] "Richtlijn Forensische Geneeskunde Postmortaal interval", Forensisch Medisch genootschap, vol. 1, no. 1, 2012.

[6] (2019, January). Thermal conductivity. Retrieved from https://en.wikipedia.org/wiki/Thermal_conductivity
[7] Mills, A.F. (1999). *Basic Heat and Mass Transfer.* Pearson Education Limited.
[8] Castellano, M., Elzer, T., Jokic, D., & Rijnders, K. (2019). Determining Thermal Conductivity of Textiles for Forensic Investigations. TU Delft.
[9] Stolk, M., Ramsey, C., Tantuo, B., & Vijayaragavan, J. (2018). Improving the ToD estimate by measuring thermal conductivity. TU Delft.
[10] Churchill, S., & Chu, H. (1975). Correlating equations for laminar and turbulent free convection from a vertical plate. International Journal of Heat and Mass Transfer, 18(11), 1323-1329.
[11] Lloyd, J., & Moran, W. R. (1974). Natural Convection Adjacent to Horizontal Surface of Various Planforms. Journal of Heat Transfer(96), 443-447.
[12] Engineering Toolbox (2019) Thermal Conductivity of common Materials and Gases, retrieved from https://www.engineeringtoolbox.com/thermal-conductivity-d_429.html
[13] Steinhart-Hart Temperature Calculator. (2016) Retrieved from https://daycounter.com/Calculators/Steinhart-Hart-Thermistor-Calculator.phtml
[14] Das, A. Alagirusamy, R. Kumar, P. (2011) Study of heat transfer through multiplayer clothing assemblies: A theoretical prediction (Vol. 11, No. 2). New Delhi, Indian Institute of Technology
[15] Arduino (2019) Arduino PWM control https://www.arduino.cc/reference/en/language/functions/analog-io/analogwrite/

Appendix I: Measuring principles

Guarded hot plate method

The guarded hot plate method is a steady-state technique used to determine the thermal conductivity of an insulating material and is commonly used in laboratories. The fact that the measurement occurs at steady state means that the clothing sample must reach a thermal equilibrium for an accurate estimate of the thermal conductivity. The method works as follows: two samples of equal dimensions are placed on either side of a heated plate, as can be seen in figure 1. Cold plates are then placed against the samples of the insulating material. This entire system is insulated, so as to prevent heat loss to the environment. Heat from the hot plate flows through the sample materials and into the cold plates. The setup is pictured below. Once equilibrium is reached, the thermal conductivity is calculated using the rise in temperature of the cold plates, which represents the heat conduction through the insulator. Thermal conductivity can be calculated using the equation given below. Variables included in formula (1) are as follows: Q is the total amount of energy supplied through the hot plate, d is the distance between the heater and the cold plate, A is the contact area of the samples with the plates and (T_{hot}-T_{cold}) represents the difference in temperature between the heat plate and the cold plates.

(1) $$k = \frac{Q * d}{A * (T_{hot} - T_{cold})}$$ [7]

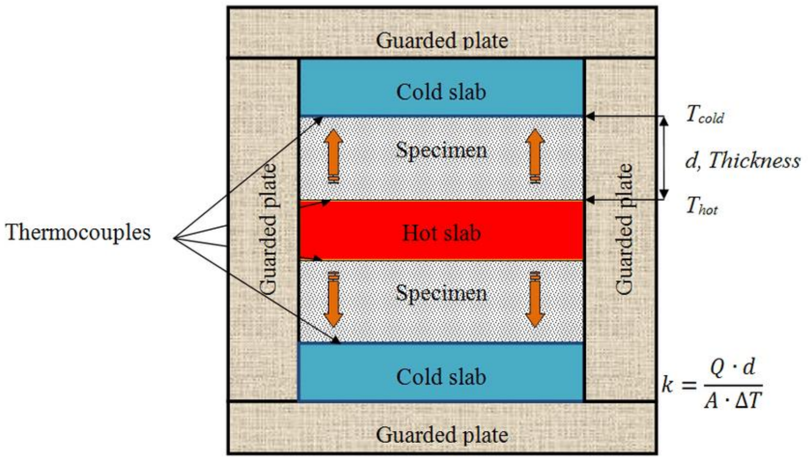


Figure 1: Schematic illustration of the guarded hot plate method. The arrows indicate thermocouples used to measure the temperature of the sample.

Hot wire method

The hot wire method is a transient technique for measuring conductivity of materials, which means that no thermal equilibrium is required for an accurate conductivity measurement. The technique consists of a wire that is heated, a sample of material with known conductivity and a sample of the textile, see figure 2. The value for the conductivity of the textile is unknown but desired. When the wire is heated, the electrical resistance of the wire changes over a specified time interval. By measuring the change of resistance the temperature change can be determined. The rate in change of temperature of the wire is dependent of the conductivity of the sample.
Two assumptions are made in the hot wire method. Firstly, the heat flow along the wire is assumed continuous and uniform along the wire. Secondly, both the sample with known and unknown conductivity are assumed to be isotropic.

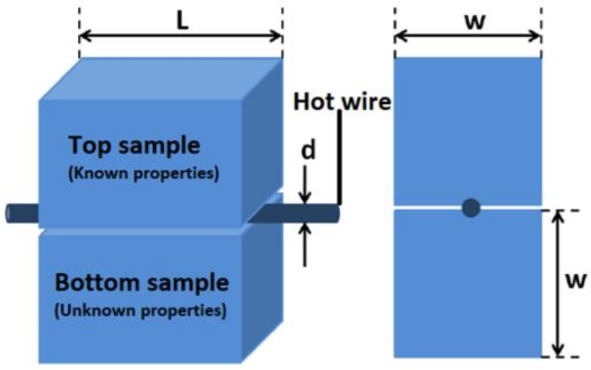


Figure 2: Setup of the hot wire method [4]

In the case that the top and bottom sample have the same unknown conductivity, the thermal conductivity of the sample can be calculated with the following formula (2):

(2) $$k = \frac{q*(\ln(t_2)-\ln(t_1))}{4\pi*(T_2-T_1)}$$ [4]

In this formula, q is the heat input per unit length of wire. T1 is the temperature of the wire after time t1 and T2:temperature of the wire after time t2

It is undesirable in the Therminus project to use two samples of unknown material, because this would require cutting larger samples off the victim's clothing. Also the design of the device could be difficult for a two sided measurement device. To this end, the formula could be altered to a form where a sample with known thermal properties is placed on one side of the hot wire.

Modified Transient Plane Source method (MTPS)

Another method is the modified transient plane source method, otherwise known as MTPS. This method uses a spiral shaped thermal resistor, which acts as both a sensor and a heating element, see figure 3. The resistor is placed between a sample of the material of which the conductivity is to be determined, and another material of which the thermal properties are known. When heat is generated by the sensor, the temperature increases, changing the resistance of the thermal resistor. The magnitude of the temperature increase depends on the amount of heat that is lost by conduction to the surrounding material.

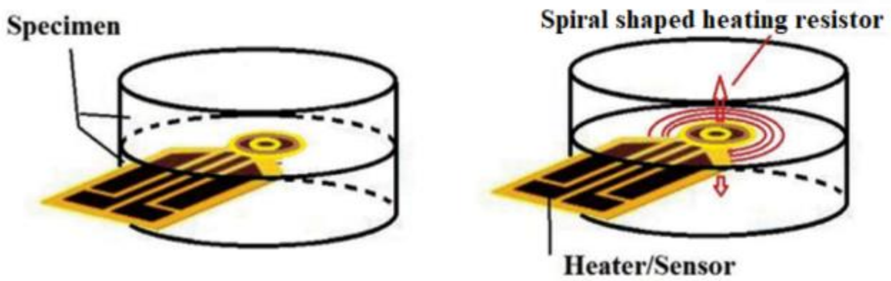


Figure 3: Schematic representation of the MTPS method

After analysing the three methods, it was determined that a variation of the guarded hot plate method is most suitable for this project. The main reason is the relative simplicity of a steady state measurement. A difference between the guarded hot plate method and the method used in this project is that the chosen working principle has only 1 cold plate and 1 material sample, with 1 directional heat-flow.

Appendix 2: Phoebe experiment

The measurement device must have a certain precision in order to be useful for forensic investigation. As stated by Olaf Dessing [1] the ToD error must be smaller than 3 percent. For determining the influence of the errors in the values for conductivity and thickness of the textile on the ToD estimation, the AMC Phoebe model is needed. The experiment done by Olaf Dessing for errors in surface thermal properties is redone for the conductivity and thickness of clothes: All values are kept constant except for the conductivity or thickness value. The difference in ToD estimation in combination with the difference in input can be translated to the maximum difference in input for a 3 percent precision. A conclusion of Olaf Dessing after his experiment was that a small conductivity value needs a higher precision than a higher value. Therefore the smallest possible conductivity, the conductivity of air, is chosen for the experiment. Experimenting with Phoebe showed that clothes with a large thickness need a higher absolute precision for the conductivity value. Therefore a thickness of 35.4mm is used in Phoebe, the thickest clothing from Appendix 14: 'Clothing thickness dataset'. The experiment also showed that the highest absolute precision for thickness is needed at a small thickness and a low conductivity value. Therefore the conductivity of air is used and the smallest thickness from the clothing database is picked, which is 0.2mm. Table 1 below shows the results of the experiment for precision determination.

*Note that table 1 is displayed on the next page in its entirety.

Table 1: Values for the accuracy value needed of conductivity and thickness for a 3% ToD estimation

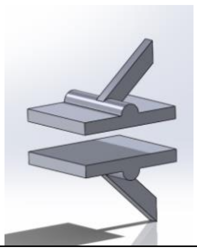
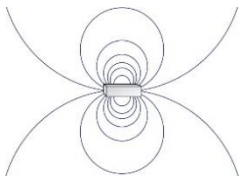
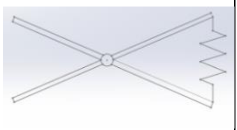
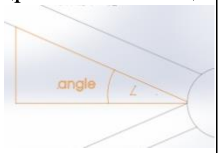
	Thermal conductivity (k) of clothes, at k=0.027W/mK, t=35.4mm	Thickness (t) of clothes, at k=0.027W/mK, t=0.2mm
Absolute accuracy required for ToD within 3% margin	0.0021W/mK	0.2mm
Relative accuracy required for ToD within 3% margin	8%	100%

As shown in the table, the maximum error for the thermal conductivity in order to have a ToD estimation with an error smaller than 3 percent is 0.0021W/mK. The largest acceptable measurement error for the thickness is 0.2mm. Note that the relative accuracy is determined for the critical required absolute accuracy, and is not necessarily the lowest percentage. For instance the relative accuracy for a textile thickness of 35.4mm at a conductivity of 0.027 is 8 percent, but the required absolute accuracy is 0.3mm. Therefore this thickness is not critical for the absolute required accuracy, but the relative accuracy has a smaller percentage.

Appendix 3: Morphological chart

In table 1 below, the morphologic chart for the design of the measurement device is shown. The chart displays various partial solutions, which are elaborated under the table.

Table 1: Morphologic chart for the design of the measurement device, in bolt the solutions chosen

Parallel measurement surfaces	Rotating surfaces 	**Linear mechanical guide**	Linear electrical guide	
Clamping	Pneumatic	Magnetic 	Mechanic spring 	**Mechanical drive**
Thickness measurement	Angle measurement (potentiometer) 	Caliper	**Linear potentiometer**	Known angle motor

Parallel measurement surfaces

- Rotating surfaces
  - Advantages
    - Simple concept
    - Robust
    - Distance measurable by angle
  - Disadvantages
    - The surfaces are held parallel by forces on the textile
    - Precisely outlining the heat-body and cold-body can be difficult
    - Most probably needs to be combined with an electric or mechanical drive
- **Linear mechanical guide**
  - Advantages
    - Robust
    - Precise outlining
    - Many distance measurement possibility's applicable
    - Modular, various drive principles applicable
  - Disadvantages
    - The guide could jam
    - Most probably needs to be combined with an electric or mechanical drive

- Linear electrical guide
    - Advantages
        - Precise outlining
        - Many distance measurement possibility's applicable, or implemented in the electrical guide
        - Does not need to be combined with an electric or mechanical drive
    - Disadvantages
        - Higher costs
        - Less robust

Clamping

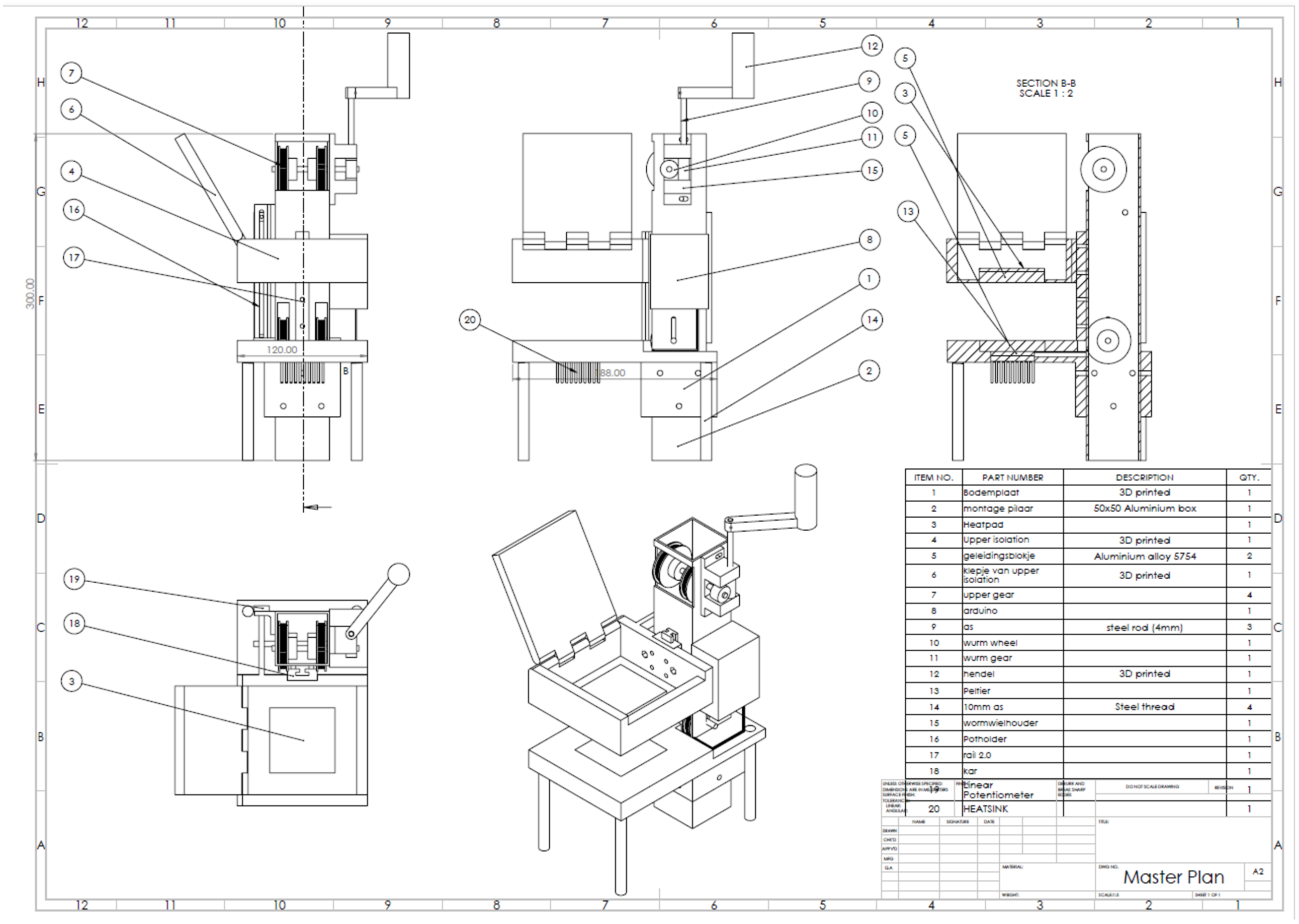
- Pneumatic
    - Advantages
        - None
    - Disadvantages
        - Too complex to implement
        - Compressed air needed at crime scene
- (Electro)magnetic
    - Advantages
        - Magnetic materials do not need external power
    - Disadvantages
        - Scales bad over distance
        - Electromagnetism needs too much electrical power
        - The force of magnetic materials cannot regulated
- Mechanic spring
    - Advantages
        - No need for electrical power or compressed air
    - Disadvantages
        - Too complex to balance for various distances
- **Mechanical drive**
    - Advantages
        - No need for electrical power or compressed air
        - Various amounts of force can be applied to measurement specimen A self-locking drive can be used
    - Disadvantages
        - The applied force cannot directly be read out

Thickness measurement

- Angle measurement
    - Advantages
        - Simple principle
    - Disadvantages
        - Small error in angle results in a large error in thickness measurement
- Caliper
    - Advantages
        - Easily applicable on mechanic systems
        - No need for electrical power
    - Disadvantages
        - Reading out of the distance is somewhat complex for the user (not just an output value on a screen)

- **Linear potentiometer**
    - Advantages
        - Easy reading out for the user by showing the distance on a screen
        - High precision
    - Disadvantages
        - More complex to implement in the device, Arduino coding needed
- Known angle motor
    - Advantages
        - High precision
        - Combination with the clamping principle
    - Disadvantages
        - More complex to implement in the device, Arduino coding needed
        - Not applicable in combination with other clamping principles

Appendix 4: Solidworks assembly drawing



ITEM NO.	PART NUMBER	DESCRIPTION	QTY.
1	Bodemplaat	3D printed	1
2	montage pilaar	50x50 Aluminium box	1
3	Heatpad		1
4	Upper isolation	3D printed	1
5	geleidingsblokje	Aluminium alloy 5754	2
6	klepje van upper isolation	3D printed	1
7	upper gear		4
8	arduino		1
9	as	steel rod (4mm)	3
10	wurm wheel		1
11	wurm gear		1
12	hendel	3D printed	1
13	Peltier		1
14	10mm as	Steel thread	4
15	wormwielhouder		1
16	Potholder		1
17	rail 2.0		1
18	kar		1
19	Linear Potentiometer		1
20	HEATSINK		1

Appendix 5: Insulation

Insulation

The warm body will be insulated from the surrounding environment. An important reason is the reduction of heat transfer to the surrounding environment which has the potential to distort the measurements. Another reason that insulation is important, to reduce the total power requirement imposed on the power supply. The total power consumption is lowered because the efficiency of the device will increase upon reducing the heat transfer to the environment. This appendix will aim to justify the chosen insulation thickness.

Shielding

The assumption that the environment merely contains surfaces that exactly match the temperature of the environment, cannot be made. The environment may also suffer from forced convection which will negatively affect the measurement accuracy by affecting the measured power consumption. Thus the device is shielded from the environment to prevent said distortions. The shield is assumed to indeed remain at the environmental temperature.

Assumptions

In an effort to estimate said energy loss, a few formulae have to be combined. A total number of five assumptions are made.

1. The total heat flow consists of radiative heat transfer from the insulation surface. Free convection from the insulation surface. And lastly, conduction from the heated element to the insulation.

2.The energy balance is valid for steady state conduction, formula (1):

(1) $$Q_{insulation} = Q_{radiation} + Q_{convection}$$

3. Material properties are homogeneous

4. All thermal conductivity constants are considered as constant over the expected temperature range ($263.15 < T < 303.15$).

5. Due to the high thermal conductivity of aluminum (147 W/mK), the temperature can be assumed as being uniformly distributed across all dimensions.

Formulae

Heat transfer due to radiation is described by Stefan-Boltzmann law: formula (2).

(2) $$Q_{radiation} = \varepsilon\sigma A({T_{surface}}^4 - {T_\infty}^4)$$

Heat transfer due to convection as proposed by Stuart W. Churchill and Humbert H.S. Chu [10] is described by (3). The Nusselt number is calculated from (4) as by J. R. Lloyd and W. R. Moran [11].

(3) $$Q_{convection} = \frac{Nu*k*A}{Lc}(T_{surface} - T_\infty)$$

(4) $$Nu = (0.65 + 0.36(\frac{Pr*g*Lc^3}{T_{estimate}*v^2})^{1/6})^2$$

$Q_{insulation}$ is calculated using formula (5) as described by Basic Heat and Mass Transfer by A.F. Mills found on page 153. When constructing the aforementioned formula, one finds that S is equal to formula (6). Table 1 will be linking abbreviations used in formula (4) to their real life interpretation and designated value. Appendix 6: 'Heat pad' also articulated the reasoning behind the necessity for more in depth analysis of the problem.

(5) $$Q_{insulation} = k_{insulation} \times S \times \Delta T$$

(6) $$S = 4\left(\frac{b*dh}{di} + 0.54(dh+b) + 0.15di\right) + \frac{b^2}{di}$$

Table 1: linking abbreviations used in formula (4) to their real life interpretation and designated value.

Abreviation	di	dh	b
Variable	Insulation Thickness	Element Thickness	Element Width
[Value] = [dimension]	[30] = [mm]	[10] = [mm]	[60] = [mm]

*Important note for formfactor equation (6): The form factor formula requires that: $b > \frac{di}{5}$, which remains valid for $b = 0.06$ and $di = 0.03$.
Note that formula (2) through (6) can be equated using the relationship proposed by assumption (3). The MATLAB script, which will be discussed shortly, is tasked with numerically solving equation (7) since it cannot be analytically evaluated, in a reasonably straightforward way.

(7) $$Q_{radiation}(T_S) + Q_{convection}(T_S) - Q_{insulation}(T_S) = 0$$

Also note that the MATLAB script utilizes the built in numerical solver "vpasolve(equation, variable)" to solve equation (7) for the surface temperature (T_S). The built in vpasolve function is likely to make an undefined error in estimating the value of (T_S) for which equation (7) holds true. The error introduced by the aforementioned function has not been evaluated nor has it been subject to further investigation.

Values

A MATLAB file has been written in order to attain the surface temperature over the expected spectrum of environmental temperatures (T_∞). For the specific geographic location of the Netherlands, T_∞ is assumed to range between -10 C° < T_∞ < 30 C°. The MATLAB script, by default, will use the aforementioned range of T_∞ to evaluate the expected surface temperatures.

Most Critical Case

In the most crucial case, a thick coat needs to be measured. As previously stated, the measurement aspect of the Therminus device will be insulated. Insulation materials which are commonly available do not differ orders of magnitude from the lowest expected thermal conductivity, 0.022 and 0.057 respectively. Thus, heat flow, through the insulation, to the environment will become more significant (more than 300 times) as the thermal conductivity of clothing becomes lower. Significant is quantified by comparing the expected relative heat loss at highest thermal conductivity ($k = 0.52$) to the expected relative heat loss at the lowest thermal conductivity ($k = 0.057$). Using the MATLAB script grants an expected relative heat loss of 0.2133% for $k = 0.52$ and 51.68% for $k = 0.057$ respectively (at T_∞= 303.15 K°). This ascertains the suspicion that the most critical case is for low thermal conductivity. The heat flow used in the measurement will evidently equal the amount of energy supplied minus the energy lost (8).

(8) $$Q_{measurement} = Q_{supplied} - Q_{insulation}$$

Whilst the statement of equation (8) seems unnecessary to the inattentive, the most critical case justifies its importance. Namely, to meet the accuracy criterion (accuracy to which the thermal conductivity is

measured) the $Q_{insulation}$ term cannot justifiably be omitted. The Arduino evaluation will therefore be required to efficaciously estimate $Q_{insulation}$.
The MATLAB script provides the expected energy loss as function of environment temperature. The Therminus device has been equipped with a calibrated environment temperature sensor. The environment temperature consequently enables an attempt at determining the value of $Q_{insulation}$. This attempt will, and regrettably so, not be flawless and thus contribute in error propagation. Repercussions of which are to be discussed in the appendix specifically dealing with error propagation. The MATLAB script provides the viewer with a set of important figures. All figures are described briefly, note that the figures are created for the most critical case in which k = 0.057 and d = 28 mm. It is imperative to note that ΔT is defined by the surface temperature from which the environment temperature is subtracted.

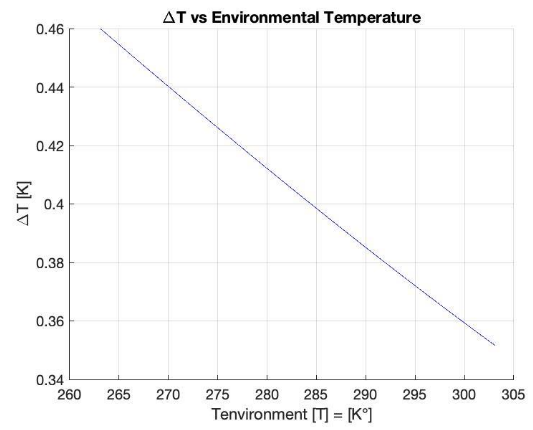


Figure 1: ΔT as function of T∞

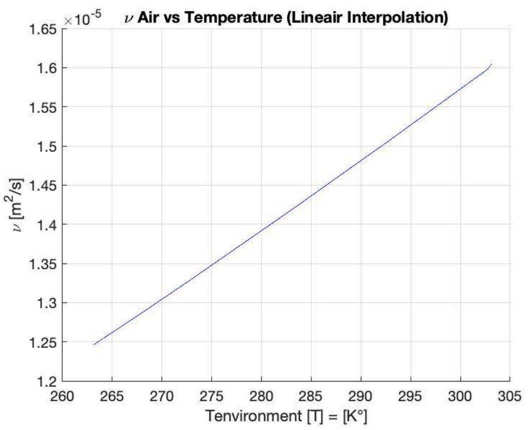


Figure 2: The interpolation used to determine what kinematic viscosity of air is applicable

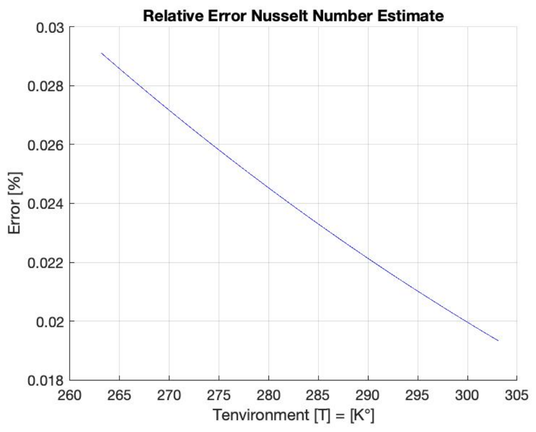


Figure 3: The relative error resulting from surface temperature estimation

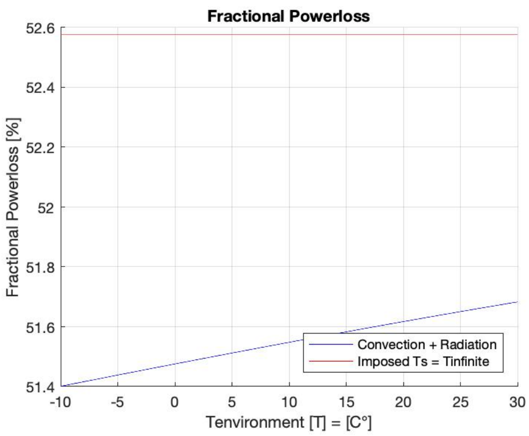


Figure 4: Fractional power loss for range of expected T∞ values

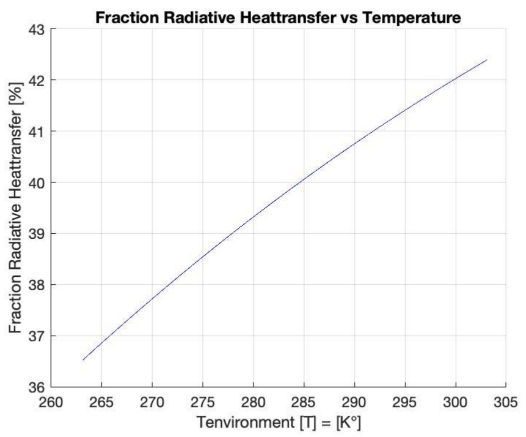


Figure 5: Fraction of total heat flow in system lost through radiative heat transfer.

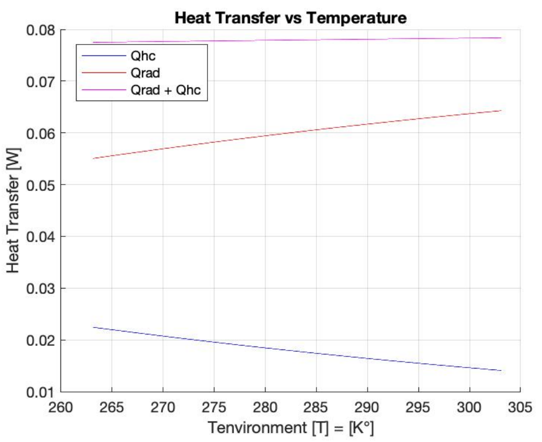


Figure 6: Total power lost to insulation as function of temperature.

Important remarks

The error caused by estimating the surface temperature in order to calculate a Nusselt number is negligible (maximum 0.0291%) in the grand scheme of things. The importance for the use of a proper heat loss model is stressed by figure 4. One can clearly tell that evaluating heat flow using formula (5) whilst using a surface temperature equal to the environment, is detrimental to the accuracy of the prediction. One must also note that the least favorable case (in which a thick coat is measured), with an environment temperature higher than 8C° is unlikely to be worn. Whilst the mentioned scenario is unlikely, the device still is required to perform sufficiently at environment temperatures up to the maximum expected environment temperature (T_∞) of 30 C°. Figure 5, in conjunction with figure 6, show the importance of modeling radiative heat transfer. The fact that total heat flow comprises for 42.41% out of radiative heat transfer, and dwarfs convective heat transfer by a factor of approximately 4.5, is rather high. Thus substantiating the statement that shielding is necessary.

Outputs from MATLAB script

```
Case 1 (unfavorable case): [k = 0.057 W/mK, d = 28.0*10^-3]
Surface Temperature delta max: 0.45984 C°. min: 0.35155 C°
Maximum Fractional Powerloss Conductivity Measurement: 51.6827% @303.15 K°
Max Heatpad Power Requirement: 0.15163 W
Fraction Heatflow Total Heatloss max: 51.6827%. min: 51.4008%
Fraction Heatflow Radiative Heatloss max: 42.4058%. min: 36.5252%
Maximum Error Qinsulation [Error, @T]: [2.8554e-03 W, @-10 C°]
For all Tinf = [Qisolation, Qmeasurement, @T]: [0.0783686    0.0732654           30]

Case 2 (optimal case): [k = 0.057 W/mK, d = 28.0*10^-3]
Surface Temperature delta max: 0.45984 C°. min: 0.35155 C°
Maximum Fractional Powerloss Conductivity Measurement: 0.21336% @303.15 K°
Max Heatpad Power Requirement: 36.7305 W
Fraction Heatflow Total Heatloss max: 0.21336%. min: 0.21097%
Fraction Heatflow Radiative Heatloss max: 0.17506%. min: 0.14992%
Maximum Error Qinsulation [Error, @T]: [2.8554e-03 W, @-10 C°]
For all Tinf = [Qisolation, Qmeasurement, @T]: [0.0783686      36.6521           30]
```

Time to Steady State

The machine needs to run for a certain amount of time before steady state is reached and the actual measurements can begin. To calculate an order of magnitude of time in which steady state is reached a

MATLAB script was written using equation 3.58 from page 169 of Mills [7] and seeing when the power loss through the insulation is the same as the heat transfer through radiation and convection on the outside of the insulation box. This formula is for an infinite body where the temperature at depth h[m] can be calculated after time t[s] when a constant temperature at the surface of the body is applied. This formula was used because only an order of magnitude of the time was needed to check if it falls between the requirements of the device. The formula used is equation (9)

(9) $$\frac{(T - T0)}{(Ts - T0)} = erfc\left(\frac{h}{\sqrt{4*alpha*t}}\right)$$

where T is the temperature of the outer surface of the insulation. T0 is the environment temperature and Ts is the temperature of the warm body, h is the thickness of the insulation in meters, alpha is the thermal diffusivity in m^2/s and t is time in seconds. Erfc is the complimentary function of the erf function which is

(10) $$\frac{2}{pi^{1.2}}\int_0^{eta} e^{-u^2}\ du$$

For MATLAB the equivalent formula on page 968 of Table B.4 from Mills is used. This is the erfc formula but rewritten for computer use.

(11) $$Erfc\ eta = (a1 * x + a2 * x^2 + a3 * x^3) * e^{-eta^2}$$

where a1 = 0.3480242, a2 = -0.0958798 and a3 = 0.7478556 as is found in Mills [7]. And $x = (1 + p * \eta)^{-1}$. Where $p = 0.47047$ a found in Mills [7] and $\eta = \frac{h}{\sqrt{4*\alpha*t}}$

According to the second MATLAB script below, using these formulas at 30mm of insulation with a thermal conductivity of 0.022 W/mK, steady state is reached after 435 seconds which is more than 7 minutes. This falls well in the production requirements where a measurement needs to be done in 30 minutes as this leaves 23 minutes to measure k.

To be on the safe side and because the calculations used in the MATLAB script are not exact formulas for this use case and only used to give an order of magnitude, would it take hours minutes or seconds, the actual measurements will be done 10 minutes after the device is turned on. Leaving 20 minutes to calculate a value for the thermal conductivity of the sample.

A second script was written to check if there is an optimum insulation thickness and the time it takes to reach steady state. With and insulation thickness of 7 mm the time to reach steady state is the shortest with 217s, figure(10). However this was not a good option for insulation thickness because the power losses through the insulation would be higher than the desired power loss through the insulation for the experiments.

Recommendations: To calculate a more precise time after which steady state is reached a different formula should be used. This formula is made for an infinite body with a constant surface temperature. While the surface temperature the insulation is not an infinite body. To calculate a more precise time after which steady state is reached a formula should be derived where the body is not infinite and the temperature at depth h should be the temperature at the outside surface of the insulation.

The MATLAB script Steady state time estimate at fixed thickness produces 3 figures, figure(7) figure(8) and figure(9). Figure(7) gives the difference in heat flow, Qdelta (W), between Qinsulation (W) and Qradiation (W) and Qconvection (W). In this graph also the time points are marked when the insulation surface first has a rise in temperature and when steady state is reached. Figure(8) shows the temperatures of the hot body, the environment and the insulation surface temperature. In the graph again the time points are marked when for the first time there is a rise in insulation surface temperature and when steady state is reached. Figure(9) is an extension on figure (7) where not only is difference in heat flow, Qdelta, plotted but also the Qinsulation, Qradiation and Qconvection. These graphs are made with a thickness of 30mm.

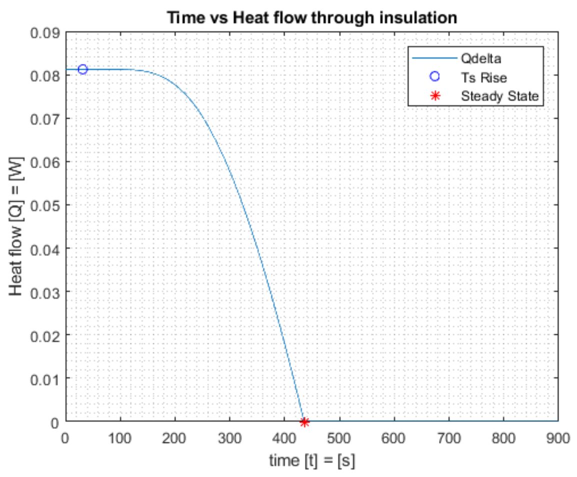


Figure 7: Qdelta vs Time, where the timepoint steady State and surface area temperature rise are marked

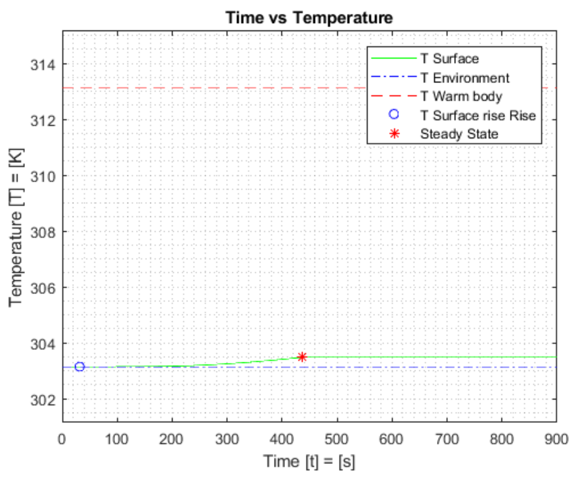


Figure 8: Time vs Temperarure, where temperatures environment, warm body and insulation surface are given, and the time points of surface temp rise and steady state are marked

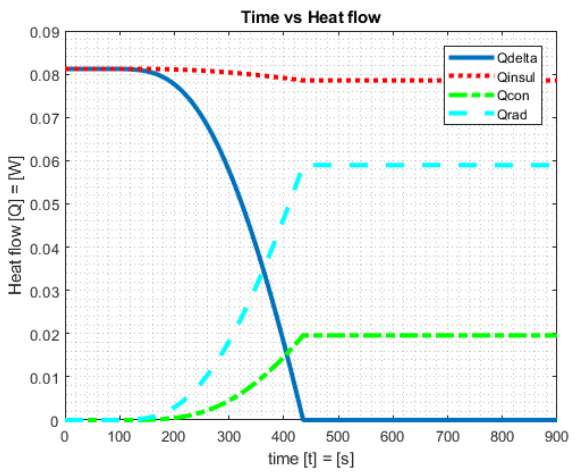


Figure 9: Time vs Heat flow, in this graph Qdelta, Qinsulation, Qconvection and Qradiation are plotted

The MATLAB script Steady state time estimate at varying thickness produces figure(10), figure(11) and figure(12). In figure(10) a line is plotted which gives the relation between how long it takes for steady state is reached and what the insulation thickness is. Here a shortest time to reach steady state can be found at 7mm insulation thickness, but looking at figure(11) is becomes clear that 7mm might not be the best choice for insulation thickness. In figure(11) a relation can be seen between thickness of insulation and the heat loss through the insulation, here is becomes clear that at 7mm there is still a too high heat loss according to the previous mentioned heat loss requirement. It is also visible that the lines flattens in the end meaning a thicker insulation does not provide an equal large reduction in heat loss. Figure(12) provides information on what the insulation surface temperature will be at different insulation thicknesses.

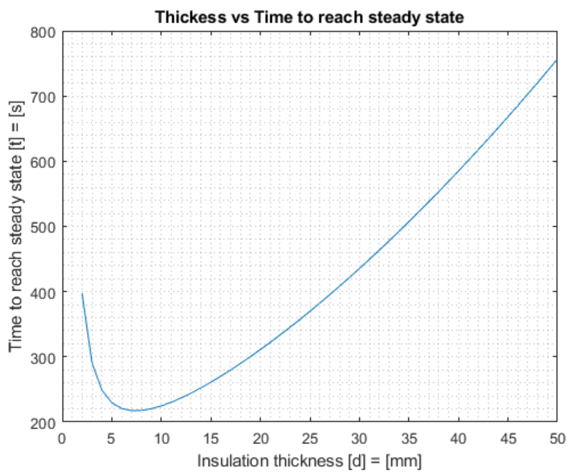


Figure 10: Insulation thickness vs time it takes for steady state is reached.

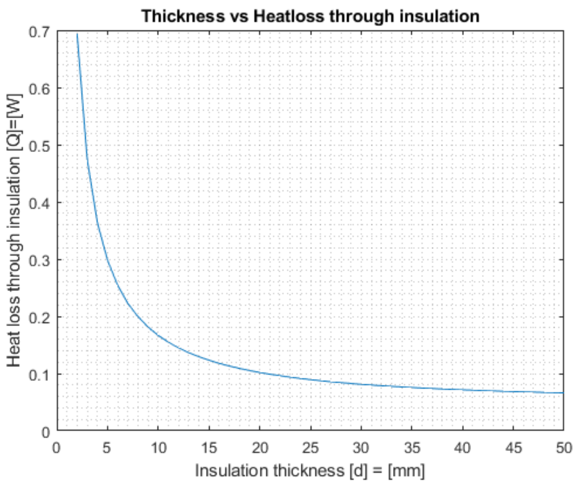


Figure 11: Insulation thickness vs heat loss through insulation.

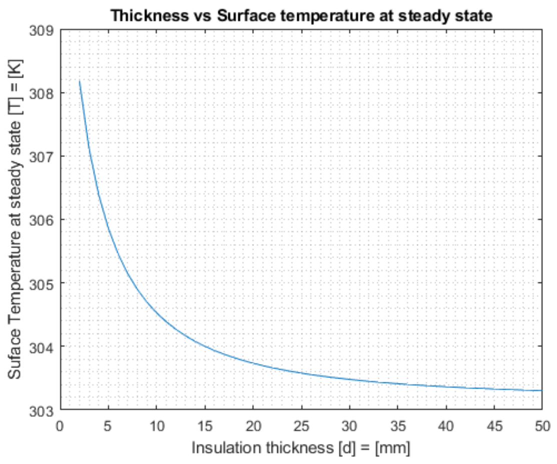


Figure 12: Insulation thickness vs surface temperature of the insulation

```
% Insulation Expected Heat Loss
% TU Delft, 05/31/2019
% Author: Sebastiaan Thomas Njio
clc
clear all
close all

%Authors remark: Note that a lot of the remarks and variables
%throughout the script are in Dutch, I would therefore recommend a Dutch
%speaking individual to evaluate the script if nescessary.

%Simulatie Specific
k_textiel = 0.057;  % Thermische geleidbaarheid textiel
d_textiel = 28.00*10^(-3);     % Dikte van het te meten textiel
di = 30*10^(-3);             % Isolatie dikte

%Variables
resolutie = 1.0* 10^(-2);
dT_meeting = 10;
Tinf_vec = 263.15:resolutie:303.15;
Telement_vec = Tinf_vec + dT_meeting;
T_estimate_vec = Tinf_vec + 0.14; %Estimate
emisivity = 0.86;   %emesivity of styrofoam

%Constants
g = 9.81;              % Valversnelling
ki = 0.022;            % Isolatie (piepschuim)
k_element = 147;       % Verwarmde en gekoelde element (5754 AlMg3)



    %Geometry
b = 60*10^(-3);     % De breedte van het verwarmde element
W = b+2*di; %Breedte oppervlak convectie
d_element = 10*10^(-3);      % Verwarmde element|
d_koud = 10*10^(-3);         % De afstand van de interface tussen textiel en
Lc = W/4; % Characteristic Length

    %Important or much occuring constants.
A = W^2;               % Surface Area on which conduction takes place
ASigma = emisivity*(W^2 + 4*(di+d_element)*W)*5.67036713*10^(-8);
%Stefanboltzmann * Area for Radiative heattransfer
S = 4*b*d_element/di + (b^2)/di + 4*0.54*d_element + 4*0.54*b + 4*0.15*di;
%Formfactor
Qmeeting = (dT_meeting) * 1/(d_element/(b*b*k_element)+d_textiel/(b*b*k_textiel)
+ d_koud/(b*b*k_element));

%Air Properties
Pr = 0.71; % Prandtl Number, considered as constant
k_air = 0.027;

%Kinematic viscosity Source: https://www.engineeringtoolbox.com/air-absolute-
kinematic-viscosity-d_601.html
vtable = [-10, -5, 0, 5, 10, 15, 20, 25, 30, 35, 40;
          12.43, 12.85, 13.28, 13.72, 14.16, 14.61, 15.06, 15.52, 15.98, 16.92,
17.88];
vtable(1,:) = vtable(1,:)+273.15;
vtable(2,:) = vtable(2,:)*10^(-6);
```

```
% Creating vectors
hc = zeros(1,length(Tinf_vec));
Ts = zeros(1,length(Tinf_vec));
Nu_error = zeros(1,length(Tinf_vec));
Ts_error = zeros(1,length(Tinf_vec));
r = zeros(1,length(vtable));
c = zeros(1,length(vtable));
dT = Ts;
counter = 1;

reg = zeros(8,length(Tinf_vec));

%Much occuring constants
C1 = (0.36^6)*Pr*g*Lc^3; C2 = k_air*A/Lc;
C3 = ki*S;

syms x

for i = 1:length(Tinf_vec)
%Read in Temperature values
Tinf = Tinf_vec(i);
Telement = Telement_vec(i);
T_estimate = T_estimate_vec(i);

beta = 1/(Tinf); %Note that this is not the real value of beta so an error is
introduced.

%Find appropriate kinematic viscosity
[r,c]=find(T_estimate> vtable(1,:));
v = vtable(2,c(end)) + (vtable(2,c(end)+1) - vtable(2,c(end)))*(T_estimate -
vtable(1,c(end)))/(vtable(1,c(end)+1)-vtable(1,c(end))); % Kinematic viscosity of
air @ 273.15K; 12.45< v <16.92

%CQhc = C2*(beta/(v^2))^(1/4);
CQhc = C1*beta/(v)^2;

%Temperature Calculation
Equation = C2*(x-Tinf)*(0.65+CQhc*(x-Tinf)^(1/6))^2 + ASigma*(x^4 - Tinf^4) +
C3*(x - Telement)==0;
Ts(1,i) = vpasolve(Equation, x);

%Error estimation
Nu_error(1,i) = 100*(((Ts(1,i)+Tinf)/(2*Tinf))^(1/3) - 1);

%Save important information
dT(1,i) = Ts(1,i) - Tinf;
Qrad = ASigma*(Ts(1,i)^4 - Tinf^4); %Radiative heattransfer Qrad = A*sigma*(T1^4-
T2^4) * Asumption: black surfaces
Qhc = C2*(Ts(1,i)-Tinf)*(0.65+(CQhc*(Ts(1,i)-Tinf))^(1/6))^2;
Qis = ki*S*(Telement-Ts(1,i));

% Registry
reg(1,i) = Tinf;
reg(2,i) = Qhc;
reg(3,i) = Qrad;
reg(4,i) = Qis;
reg(5,i) = v;    % Display interpolation kinematic viscosity of air
reg(6,i) = (Qhc+Qrad)/(Qmeeting + Qhc + Qrad)*100; %Fractional Powerloss
reg(7,i) = (Qrad/(Qmeeting + Qhc + Qrad))*100;
reg(8,i) = CQhc/(0.36^6)*1/beta*1/(Ts(1,i));

if i/length(Tinf_vec)>counter/100
    counter = counter + 1;
    disp(['Progression: ',num2str(counter),'%'])
end
end



%Data manipulation
Qlossfr = reg(6,:); %Combined fractional heatloss to environment
Qlossradfr = reg(7,:); %Fractional Heatloss due to radiation
% Alternative heatloss
Qdotloss_sfr = ki*S*dT_meeting/(Qmeeting + ki*S*10)*ones(1,length(reg(5,:)))*100;
%Gesimplificeerd model fractional Heatloss
%% Figures and Display outputs



%%Figures
figure()
grid on; hold on
x0 = 100; y0 = 600; height = 300; width = 400;
set(gcf,'units','points','position',[x0,y0,width,height]);
title('\DeltaT vs Environmental Temperature');xlabel(['Tenvironment [T] =
[K',char(176),']']); ylabel('\DeltaT [K]');
plot(Tinf_vec, dT,'-b')

figure()                    %Interpolation kinematic viscosity
grid on; hold on
x0 = 120+width; y0 = 600;
set(gcf,'units','points','position',[x0,y0,width,height]);
title('\nu Air vs Temperature (Lineair Interpolation)');xlabel(['Tenvironment [T]
= [K',char(176),']']); ylabel('\nu [m^2/s]');
plot(Tinf_vec, reg(5,:),'-b')

figure()                    %Nusselt number error
grid on; hold on
x0 = 140+2*width; y0 = 600;
set(gcf,'units','points','position',[x0,y0,width,height]);
title('Relative Error Nusselt Number Estimate');xlabel(['Tenvironment [T] =
[K',char(176),']']); ylabel('Error [%]');
plot(Tinf_vec, Nu_error,'-b')

figure()                    %Fraction Powerloss
grid on; hold on
x0 = 100; y0 = 520-height;
set(gcf,'units','points','position',[x0,y0,width,height]);
title('Fractional Powerloss');xlabel(['Tenvironment [T] = [C',char(176),']']);
ylabel('Fractional Powerloss [%]');
plot(Tinf_vec-273.15, Qlossfr,'-b', Tinf_vec-273.15, Qdotloss_sfr,'-r')
legend('Convection + Radiation','Imposed Ts = Tinfinite','Location','southeast')

figure()                    %Radiation Fraction Powerloss
grid on; hold on
x0 = 120+1*width; y0 = 520-height;
set(gcf,'units','points','position',[x0,y0,width,height]);
title('Fraction Radiative Heattransfer vs Temperature');xlabel(['Tenvironment [T]
= [K',char(176),']']); ylabel('Fraction Radiative Heattransfer [%]');
```

```
plot(Tinf_vec, Qlossradfr,'-b')

figure()                        %Heat Transfer Plot
grid on; hold on
x0 = 140+2*width; y0 = 520-height;
set(gcf,'units','points','position',[x0,y0,width,height]);
title('Heat Transfer vs Temperature');xlabel(['Tenvironment [T] =
[K',char(176),']']); ylabel('Heat Transfer [W]');
plot(Tinf_vec, reg(2,:),'-b', Tinf_vec, reg(3,:),'-r',Tinf_vec,reg(2,:) +
reg(3,:),'-m')
legend('Qhc','Qrad','Qrad + Qhc','Location','northwest')

%Display values
disp(['Surface Temperature delta max: ', num2str(max(dT)),' C',char(176),'. min:
', num2str(min(dT)),' C',char(176)])
[M,I] = max(reg(6,:)); % Finding max fractional powerloss @ Temperature I
disp(['Maximum Fractional Powerloss Conductivity Measurement: ',num2str(M),'%', '
@',num2str(Tinf_vec(1,I)),' K',char(176)])
disp(['Heatpad Power Requirement: ',num2str(Qmeeting+max(reg(4,:))),' W'])
disp(['Fraction Heatflow Total Heatloss max: ', num2str(max(reg(6,:))), '%','.
min: ', num2str(min(reg(6,:))), '%'])
disp(['Fraction Heatflow Radiative Heatloss max: ', num2str(max(reg(7,:))),
'%','. min: ', num2str(min(reg(7,:))), '%'])
```

```
%% Finding Steady state conduction through insulation %%
close all
clear all
clear
clc

%% Variables
Sa = 60;            % mm, Length and Width Aluminium
Ha = 10;            % mm, height aluminium
Thi = 30;           % mm, thickness insulation
Te = 303.15;        % K, environment temperature
Td = 10;            % K, Delta T between warm body and environment
time = 900;         % s, time
timestep = 0.01;    % s, timesteps

%% Constants

kp = 0.022;         % W/mK of polystyrene
ka = 0.0252;        % W/mK of air at 275K
cpp = 1210;         % J/kgK of polystyrene
cpa = 1008.5;       % K/kgK of Air at 275K
rhop = 45;          % kg/m3 of polystyrene
rhoa = 1.288;       % kg/m3 or Air at 275K
epla = 0.84;        % emmitance white plastic
sigma = 5.67*10^-8; % W/m2K4, Stefan-Boltzmann constant
Beta = 1/Te;        % 1/K
va = 15.98*10^-6;   % m2/s of Air at 275K
Pr = 0.71;          % - of Air at 275K
g = 9.81;           % m/s2 gravity

% erfc constants (Mills page 968 table B.4)
a1 = 0.3480242;
a2 = -0.0958798;
a3 = 0.7478556;
p = 0.47047;

%% Parameters
% Dimensions
la = Sa/1000;                   % m, length and width of aluminium
ha = Ha/1000;                   % m, height aluminium
tin = Thi/1000;                 % m, thickness insulation
l = (Sa+2*Thi)/1000;            % m, length and width insulation
h = (Ha+Thi)/1000;              % m, height insulation
A1 = l*l;                       % m2, top area insulation
A2 = h*l;                       % m2, side area insulation

%Time
t = [0.01: timestep: time]';    % s, time

% Thermal diffusivity
alphap = kp/(rhop*cpp);         % m2/s of polystyrene
alphaa = ka/(rhoa*cpa);         % m2/s of Air at 275

%Heat transfer
S = (la*la)/tin + 4* ((la*ha)/tin) + 4*0.54*ha + 4*0.54*la + 4*0.15*tin; %Shape
factor

%Temperature
Tb = Te+Td;                     % K, temperature warm body
```

```
%% calculations
%Counter
j = 1;
q = 1;
w = 1;
z = 0;

% Space allocation
n = zeros(size(t));
x = zeros(size(t));
erfc = zeros(size(t));
Ts = zeros(size(t));
Ra = zeros(size(t));
Nu = zeros(size(t));
hc = zeros(size(t));
Qcon = zeros(size(t));
Qrad = zeros(size(t));
Qinsul = zeros(size(t));
Qdelta = zeros(size(t));


%Loop
for i=1:length(t)
    n(j, 1) = h/sqrt((4*alphap*t(j)));
    x(j, 1) = 1/(1+p*n(j));
    erfc(j, 1) = (a1*x(j)+a2*(x(j)^2)+a3*(x(j)^3))*exp(-(n(j)^2));

    if z == 0
        Ts(j, 1) = erfc(j)*(Tb-Te)+Te;
    else
        Ts(j,1) = Ts(stop);
    end

    Ra(j, 1) = Beta*(Ts(j)-Te)*g*(l^3)/(va*alphaa);
    Nu(j, 1) = (0.65+0.36*(Ra(j)^(1/6)))^2;
    hc(j, 1) = ka*Nu(j)/l;
    Qcon(j, 1) = (Ts(j)-Te)*A1*hc(j) + (Ts(j)-Te)*A2*hc(j)*4;
    Qrad(j, 1) = A1*epla*sigma*((Ts(j)^4)-(Te^4))+4*A2*epla*sigma*((Ts(j)^4)-
(Te^4));
    Qinsul(j, 1) = kp*S*(Tb-Ts(j));
    Qdelta(j, 1) = Qinsul(j) - (Qcon(j)+Qrad(j));
    if Qdelta(j, 1) < 0
        Qdelta(j,1) = 0;
        stop = find(Qdelta,1,'last');
        z = 1;
        if w == 1
            tss = j;
            w = 3;
        end
    end
    if Qdelta(j, 1) < Qinsul(j)
        if q == 1
            tpen = j;
            q = q+2;
        end
    end


    j = j+1;
end
```

```
%% Figures
graph_mm = 2;
Te = ones(size(t))*Te;
Tb = ones(size(t))*Tb;
% 1
figure(1)
plot(t, Qdelta)
hold on
plot(t(tpen),Qdelta(tpen),'ob',t(tss),Qdelta(tss),'*r')
str1 = num2str(t(tpen));
str2 = num2str(t(tss));
text(t(tpen),Qdelta(tpen),str1)
text(t(tss),Qdelta(tss),str2)
xlabel('time s')
ylabel('Heat flow W')
legend('Qdelta','Ts Rise','Steady State')

% 2
figure(2)
plot(t,Ts,t,Te,t,Tb)
xlim([0 time])
ymin = Te(1) - graph_mm;
ymax = Tb(1) + graph_mm;
ylim([ymin ymax])
xlabel('Time s')
ylabel('Temperature K')
hold on
plot(t(tpen),Ts(tpen),'ob',t(tss),Ts(tss),'*r')
legend('Ts','Te','Tb','Ts Rise','Steady State')
text(t(tpen),Ts(tpen),str1)
text(t(tss),Ts(tss),str2)

% 3
figure(3)
plot(t, Qdelta, t, Qinsul, t, Qcon, t, Qrad)
legend('Qdelta','Qinsul','Qcon','Qrad')
xlabel('time s')
ylabel('Heat flow W')
```

```
%% Comparing the time to reach steady state with different insulation
thicknesses %%
close all
clear all
clear
clc

%% Variables
Sa = 60;                % mm, Length and Width Aluminium
Ha = 10;                % mm, height aluminium
Te = 303.15;            % K, environment temperature
Td = 10;                % K, Delta T between warm body and environment
time = 900;             % s, time
timestep = 0.01;        % s, timesteps

%% Constants

kp = 0.022;             % W/mK of insulation
ka = 0.0252;            % W/mK of air at 275K
cpp = 1210;             % J/kgK of polystyrene
cpa = 1008.5;           % K/kgK of Air at 275K
rhop = 45;              % kg/m3 of polystyrene
rhoa = 1.288;           % kg/m3 or Air at 275K
epla = 0.84;            % emmitance white plastic
sigma = 5.67*10^-8;     % W/m2K4, Stefan-Boltzmann constant
Beta = 1/Te;            % 1/K
va = 15.98*10^-6;       % m2/s of Air at 275K
Pr = 0.71;              % - of Air at 275K
g = 9.81;               % m/s2 gravity


% erfc constants (Mills page 968 table B.4)
a1 = 0.3480242;
a2 = -0.0958798;
a3 = 0.7478556;
p = 0.47047;

%% Parameters
% Dimensions
Thi = [2:1:50]';                    % mm, thickness insulation
la = Sa/1000;                       % m, length and width of aluminium
ha = Ha/1000;                       % m, height aluminium


%Time
t = [0.01: timestep: time]';        % s, time

% Thermal diffusivity
alphap = kp/(rhop*cpp);             % m2/s of polystyrene
alphaa = ka/(rhoa*cpa);             % m2/s of Air at 275

%Temperature
Tb = Te+Td;                         % K, temperature warm body



%% calculations
% counters
j = 1;
q = 1;
```

```
w = 1;
u = 1;
z = 0;

% Space allocation
% Thickness
tin = zeros(size(Thi));
l = zeros(size(Thi));
h = zeros(size(Thi));
S = zeros(size(Thi));
A1 = zeros(size(Thi));
A2 = zeros(size(Thi));
stop = zeros(size(Thi));
Tscheck = zeros(size(Thi));
Qdeltmm = zeros(size(Thi));

%Time
n = zeros(size(t));
x = zeros(size(t));
erfc = zeros(size(t));
Ts = zeros(size(t));
Ra = zeros(size(t));
Nu = zeros(size(t));
hc = zeros(size(t));
Qcon = zeros(size(t));
Qrad = zeros(size(t));
Qinsul = zeros(size(t));
Qdelta = zeros(size(t));

%The Loop
for counter = 1:length(Thi)
    tin(u, 1) = Thi(u)/1000;                % m, thickness insulation
    l(u, 1) = (Sa+2*Thi(u))/1000;           % m, length and width insulation
    h(u, 1) = (Ha+Thi(u))/1000;             % m, height insulation
    A1(u, 1) = l(u)*l(u);                   % m2, top area insulation
    A2(u, 1) = h(u)*l(u);                   % m2, side area insulation
    S(u, 1) = (la*la)/tin(u) + 4* ((la*ha)/tin(u)) + 4*0.54*ha + 4*0.54*la +
4*0.15*tin(u); %Shape factor
    n = zeros(size(t));
    x = zeros(size(t));
    erfc = zeros(size(t));
    Ts = zeros(size(t));
    Ra = zeros(size(t));
    Nu = zeros(size(t));
    hc = zeros(size(t));
    Qcon = zeros(size(t));
    Qrad = zeros(size(t));
    Qinsul = zeros(size(t));
    Qdelta = zeros(size(t));
    j = 1;

    for i=1:length(t)

        n(j, 1) = h(u)/sqrt((4*alphap*t(j)));
        x(j, 1) = 1/(1+p*n(j));
        erfc(j, 1) = (a1*x(j)+a2*(x(j)^2)+a3*(x(j)^3))*exp(-(n(j)^2));

        if z == 0
            Ts(j, 1) = erfc(j)*(Tb-Te)+Te;
        else
```

```
            Ts(j,1) = Ts(stop(u));
        end

        Ra(j, 1) = Beta*(Ts(j)-Te)*g*(l(u)^3)/(va*alphaa);
        Nu(j, 1) = (0.65+0.36*(Ra(j)^(1/6)))^2;
        hc(j, 1) = ka*Nu(j)/l(u);
        Qcon(j, 1) = (Ts(j)-Te)*A1(u)*hc(j) + (Ts(j)-Te)*A2(u)*hc(j)*4;
        Qrad(j, 1) = A1(u)*epla*sigma*((Ts(j)^4)-
(Te^4))+4*A2(u)*epla*sigma*((Ts(j)^4)-(Te^4));
        Qinsul(j, 1) = kp*S(u)*(Tb-Ts(j));
        Qdelta(j, 1) = Qinsul(j) - (Qcon(j)+Qrad(j));
        Qdeltmm(u, 1) = Qdelta(1);

        if Qdelta(j, 1) < 0
            Qdelta(j,1) = 0;
            stop(u,1) = find(Qdelta,1,'last');
            z = 1;
            Tscheck(u,1) = Ts(stop(u));
        end

        j = j+1;
    end

    z = 0;
    u = u+1;

end

%% Figures
figure(1)
stopt = stop/(1/timestep);
plot(Thi,stopt)
xlabel('Insulation thickness mm')
ylabel('Time to reach steady state s')
figure(2)
plot(Thi,Qdeltmm)
xlabel('Insulation thickness mm')
ylabel('Heat loss through insulation W')
figure(3)
plot(Thi,Tscheck)
xlabel('Insulation thickness mm')
ylabel('Suface Temperature at steady state K')
```

Appendix 6: Heat pad

Heat Pad Requirements

The Therminus device has to fulfill a few requirements. A key requirement is the capability of measuring a range of thermal conductivity values. The thermal aspect from measurement depends on a few design parameters, namely: surface area, thickness, thermal conductivity and temperature difference. As stated before, the relationship is described by Fourier's law of thermal conductivity (1).

(1) $$\frac{\partial Q}{\partial t} = -k \oiint_S \nabla T \, dS$$

The values for thermal conductivity that the Therminus device should be able to measure range from 0.52 on the low end, and 0.057 [14] on the high end of the spectrum. Data based on a sample size of 25 T-shirts, Appendix 14: 'Clothing thickness dataset' suggests that 99.73% of T-shirts is expected to be at least 0.44 mm thick. Thus suggesting a minimum thermal insulance of $[R_{th}] = 8.46*10^{-04}$ $[m^2K/W]$. Again assuming coat thickness is normally distributed, in 95.45% of cases, coats will be thinner than 27.37 mm. The upper bound for thermal insulance thus becomes $[R_{th}] = 4.80*10^{-02}$ $[m^2K/W]$. From Fourier's law of thermal conductivity, a design index can be made, equation (2), from which one can deduct that the head pad is required to have a minimum $\frac{Q}{A} > 11.8*10^3$. The chosen heat pad has an index $(\frac{Q}{A})$ of $13*10^3$ and will thus be able to supply sufficient heat.

(2) $$\frac{Q}{A} = \frac{\Delta T}{R_{th}}$$

Due to cleanability requirements imposed on the Therminus device, temperature sensors will only be placed in places that, during its lifecycle, will be unaffected by muck from the environment. Therefore the temperature difference will only be measured on the hot and cold body surface temperatures. The monitored cold and hot body surfaces will be the ones opposing the clothing surfaces. When considering the complete device in steady state condition, a certain temperature distribution is to be expected. Figure 1 shows the worst case expected temperature distribution as function of cross sectional distance measured from the heat pad. Figure 2 shows other extreme case in which a well insulating coat is measured.

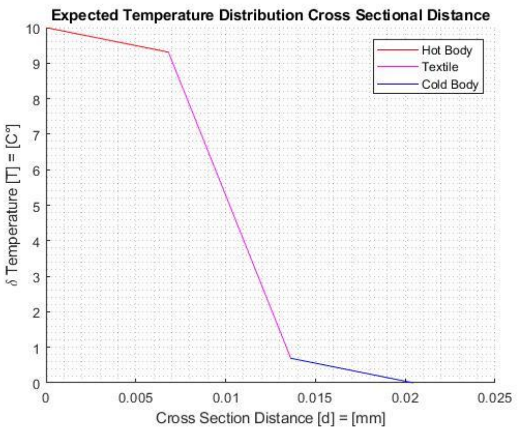


Figure 1: Temperature distribution as function of cross sectional distance for shirt.

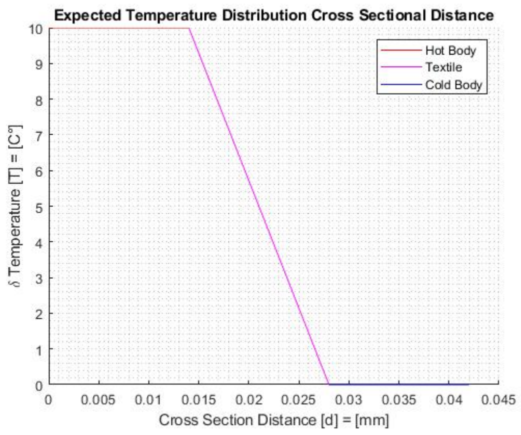


Figure 2: Temperature distribution as function of cross sectional distance for coat.

A fractional power loss estimate is made by this script. The fractional power loss (Q_{frac}) is defined by two quantities and is defined according to formula (3).

(3) $$Q_{frac} = \frac{Q_{insulation}}{Q_{measurement} + Q_{insulation}}$$

Firstly quantity being the heatflux that is nescessary for carrying out the thermal conductivity measurement: $Q_{measurement}$. The second being the heat that is lost through the insulation: $Q_{insulation}$, which in an ideal case, would be zero for several reasons. The main reason is that when said heat flux is zero, no error can be made in estimating said heat flux.

(4) $$Q_{insulation} = k_{insulation} \times S \times \Delta T$$

$Q_{insulation}$ is calculated using formula (4) as described by [7]. When constructing the aforementioned formula, one finds that S is equal to formula (5). Table 1 will be linking abbreviations used in formula 3 to their real life interpretation and designated value.

(5) $$S = 4\left(\frac{b*dh}{di} + 0.54(dh + b) + 0.15di\right) + \frac{b^2}{di}$$

Table 1: linking abbreviations used in formula 3 to their real life interpretation and designated value.

Abbreviation	di	dh	b
Variable	Insulation Thickness	Element Thickness	Element Width
[Value] = [dimension]	[30] = [mm]	[10] = [mm]	[60] = [mm]

The MATLAB script ("Temperature Distribution Therminus Device") which is provided alongside this Appendix, uses formula (4) and (5) (in conjunction with other common formulae) to estimate fractional heat loss as described by formula (3). Formulae (4) and (5) are specifically mentioned because they inherit an error caused by the assumption that $\Delta T = 10$. In other words, an error is caused by the assumption that the surface temperature of the insulation will be exactly equal to the environment temperature. This will in practice not be the case, thus an unknown error is caused by this assumption. Appendix 12: 'Heat loss insulation uncertainty' will not suffer from this error as the surface temperature will not be imposed. Please reference Appendix 12 for more detail.
Important information that is output from the script is displayed in table 2. It is imperative to stress the fact that the fractional power loss is a mere estimate made with an unrealistic assumption. Said unrealistic assumption being that the surface temperature of the insulation is exactly equal to the environment temperature, as stated in Appendix 5: 'Insulation'.

Table 2: important information that is output by the provided MATLAB script.

Case	Corresponding to figure (1)	Corresponding to figure (2)
Input values	k_textiel = 0.52; d_textiel = 0.41*$10^{(-3)}$	k_textiel = 0.057; d_textiel = 22*$10^{(-3)}$
Output	Required Power: 36.7333	Required Power: 0.17446
	Fractional Powerloss: 0.22112%	Fractional Powerloss: 46.5563%
	T1: 10 C°	T1: 10 C°
	T2: 9.3074 C°	T2: 9.9982 C°
	T3: 0.69259 C°	T3: 0.0017619 C°
	T4: 1.2212e-15 C°	T4: 1.4244e-15 C°

MATLAB Code (Appendix 6: Heat pad)

```
% Temperature Distribution Therminus Device
% TU Delft, 05/31/2019
% Author: Sebastiaan Thomas Njio

clc
clear all
close all

%Textiel waardes
k_textiel = 0.057;          % Expected values: 0.057<k<0.52
d_textiel = 22.00*10^(-3);   % Kleding dikte
% Voor een maximale

%Formules hitte verlies
% Qdot = k*S*dT
To = 0;                     % Omgevingstemperatuur
dT = 10;                     % Temperatuur delta
b = 60*10^-3;                 % De breedte van het verwarmde element met T1

% Diktes
```

```
d_element = 10*10^(-3);     % Verwarmde element
di = 30*10^(-3);            % Isolatie
d_koud = 10*10^(-3);        % De afstand van de interface tussen textiel en
                        % koude blok tot waar de temperatuur meting is.

% Thermische geleidbaarheid
ki = 0.022;                 % Isolatie (piepschuim)
k_element = 147;            % Verwarmde element (Alu 5754 bij 20graden c)

% Much Occuring Constants
A = b*b;
% Berekeningen
S = 4*b*d_element/di + (b^2)/di + 4*0.54*d_element + 4*0.54*b + 4*0.15*di; %Formfactor

T1 = To+dT; % Temperatuur heatpad

% Vermogens
Qdotloss = ki*S*dT; % Dit is het vermogen wat verloren raakt door de isolatie
Qdotmeeting = (T1 - To)  *  1/(d_element/(A*k_element)+d_textiel/(A*k_textiel) +
d_koud/(A*k_element));

Qdotheatpad = Qdotloss + Qdotmeeting;
% Temperature Calculations
Unknown = 0; % Dit is de hoeveelheid warmte die via de zijkanten van het textiel wegvloeit.

T2 = T1 - d_element/(k_element*A)*Qdotmeeting;
T3 = T2 - d_textiel/(k_textiel*A)*Qdotmeeting;
T4 = T3 - d_koud/(k_element*A)*Qdotmeeting; %Check if indeed the T4 ~ 0

% Displaying all outputs
disp(['Required Power: ', num2str(Qdotheatpad)])
disp(['Fractional Powerloss: ', num2str(Qdotloss/Qdotheatpad*100),'%'])

disp(['T1: ',num2str(T1),' C',char(176)])
disp(['T2: ',num2str(T2),' C',char(176)])
disp(['T3: ',num2str(T3),' C',char(176)])
disp(['T4: ',num2str(T4),' C',char(176)])

%Plotting figure
dTot = linspace (0, d_element + d_koud + d_textiel, 4);
Temperatures = [T1, T2, T3, T4];

figure()
hold on; grid minor
title('Expected Temperature Distribution Cross Sectional Distance'); xlabel('Cross Section Distance [d]
= [mm]'); ylabel(['\delta Temperature [T] = [C',char(176),']']);
plot(dTot(1,[1, 2]), Temperatures(1,[1, 2]),'-r',dTot(1,[2, 3]), Temperatures(1,[2, 3]),'-m',dTot(1,[3, 4]),
Temperatures(1,[3, 4]),'-b')
legend('Hot Body','Textile','Cold Body','Location','northeast')
```

Appendix 7: Electrical circuit

Two drawings have been made for the electrical circuit. The first drawing, figure 1, displays the circuit for the prototype. Please note that the figures to which this appendix refers are too big to fit appropriately inside the text. Throughout this Appendix, both figure 1 and 2 are of great relevance, which are provided at the end of the appendix. Making use of three external power supplies and thus not meeting the requirement of having a mobile device as this is dependent on power sockets. The Arduino is powered by a 12V adapter connected to a power socket and the peltier element and heat pad are both connected to a PSU of which the voltage and amperage can be regulated. This was done to test and calibrate the device.

Ideally the device would be made as is in the second drawing, figure 2. Making use of batteries in series or a bigger battery. For this to work voltage regulators need to be implemented between the power supply and the Arduino, the peltier element and the heating pad. Because each run on different voltages. The total power draw of the circuit amount to about 100W from the various devices in the circuit. The batteries used for this circuit needs to be able to deliver amount of power for the duration of the testing and ideally be not too big or heavy to keep the device portable. Two examples of such kind of batteries would be those used in electric bikes or the ones used by photographers to power their flashes.

Both circuits have in common the peltier element (PE), the heating pad (HP), the Arduino, three different temperature sensors, one for the cold body temperature (CBT) one for hot body temperature (WBT) and one for the environment temperature (ET) and a linear potentiometer(LP). A fan is used in the system to cool the heatsink connected to the peltier element.

In the drawings all positive wires are coloured red, all ground wires are coloured black. The orange wires coming from the sensors are being read by the Arduino at the analog ports. The purple wires going to the mosfets connected to the peltier element and heatpad are used for pulse width modulation (PWM) control.

PWM control by digital pins 5 and 6 and the mosfets are connected to these pins to control the peltier element and the heatpad. These pins were chosen because they are timed at 980Hz [15] and have an 8 bit resolution meaning 256 steps, while the other pwm pins of the Arduino are timed at 490Hz and 8 bit resolution. Using pins 5 and 6 means that 1/980Hz = 0.001s = 1ms is divided in 256 parts, every part being equal to 1ms/256 = 0.004ms = 4µs. Which means that the PWM control can be set from 0 to 255 where every step adds 4µs to the width of the signal. The signal gives the full voltage for the amount of time the signal is width and 0 after that until this repeats again at the next ms.

The sensors are connected to the digital pins on the Arduino to give them voltage. Each sensor has a specific pin. Warm body heat sensor pin 2. Cold body heat sensor pin 3. Environment heat sensor on pin 4. Potentiometer on pin 7. They are in the same order connected to the analog pins on the Arduino to read the values. Warm body hear sensor pin A0. Cold body heat sensor pin A1. Environment heat sensor pin A2. Potentiometer pin A3.

The analog pins on the Arduino read a voltage range from 0V to 5V with 10-bit resolution, which is 1024 steps. Which means every step corresponds to a 0.005V increase. Creating an as large as possible voltage difference between the maximum and the minimum of the sensor increases the accuracy of the sensor. Due to the added resistance the voltage range over the sensor is always smaller than 5 volt. For specific voltage ranges corresponding to the temperature or potentiometer respectively, please consult Appendix 10: 'Thermistor Uncertainty' or Appendix 11: 'Linear Potentiometer Uncertainty'. But the resistor can be chosen so that the difference is as large as possible for that specific sensor. To ensure the biggest range of voltage that can be measured an ideal resistor needs to be calculated and placed in the circuit. The general formula for the output voltage read by the Arduino for each sensor is formula (1);

(1) $$V_{out} = \frac{V_{in} * R_{sensor}}{R + R_{sensor}}$$

Here Vout is what is measured by the Arduino, R_{sensor} is what the resistance is of the sensor, R is the added resistance in the circuit and V_{in} is the voltage the Arduino puts in the circuit. The minimum and maximum resistance the sensor becomes is known via the datasheet of the sensor. The output voltage between these two resistances needs to be as large as possible to get the best sensitivity in the measurement.

$$\Delta V = \frac{V_{in} * R_{sensormax}}{R + R_{sensormax}} - \frac{V_{in} * R_{sensormin}}{R + R_{sensormin}} \tag{2}$$

In formula (2) ΔV is the difference in voltage output between maximal sensor resistance and minimal sensor resistance. In this formula V_{in} is known, $R_{sensormax}$ is known, $R_{sensormin}$ is known. What is needed is the optimal R for the largest ΔV.

This is done by calculating the derivative of the formula and setting it to zero.

$$\frac{d\Delta V}{dR} = \frac{Vin*Rsensormax^2*Rsensormin - Vin*Rsensormax*R^2 - Vin*Rsensormax*Rsensormin^2 + Vin*Rsensormin*R^2}{\left(Rsensormax*Rsensormin + Rsensormax*R + Rsensormin*r + R^2\right)^2} = 0 \tag{3}$$

Which means that the denominator needs to be set to zero. Resulting in the following equation, formula (4), and simplifying it. Rewriting gives formulas (5) and (6).

$$Vin * Rsensormax^2 * Rsensormin - Vin * Rsensormax * R^2 - Vin * Rsensormax * Rsensormin^2 + Vin * Rsensormin * R^2 = 0 \tag{4}$$

$$=$$

$$R = \sqrt{\frac{Vin * Rsensormax * Rsensormin^2 - Vin * Rsensormax^2 * Rsensormin}{Vin * (Rsensormin - Rsensormax)}} \tag{5}$$

$$=$$

$$R = \sqrt{Rsensormax * Rsensormin} \tag{6}$$

This formula (6) is put in a MATLAB script. In this MATLAB script the formula gets the minimum and maximum resistance the sensor can become and the output of the MATLAB script is the resistance that should be used to give the biggest voltage difference the Arduino can read. The resistances giving the biggest voltage difference are:
Potentiometer = 438 Ω
Temperature sensor for the hot body = 6.324 kΩ
Temperature sensor for the cold body = 12.64 kΩ
Temperature sensor for environment = 1.9 kΩ
The resistances used for the MOSFET's are only there to prevent short circuiting for which a resistance of 10kΩ is used in both cases. The fan is connected to the 5V pin of the Arduino and is operational at constant speed.

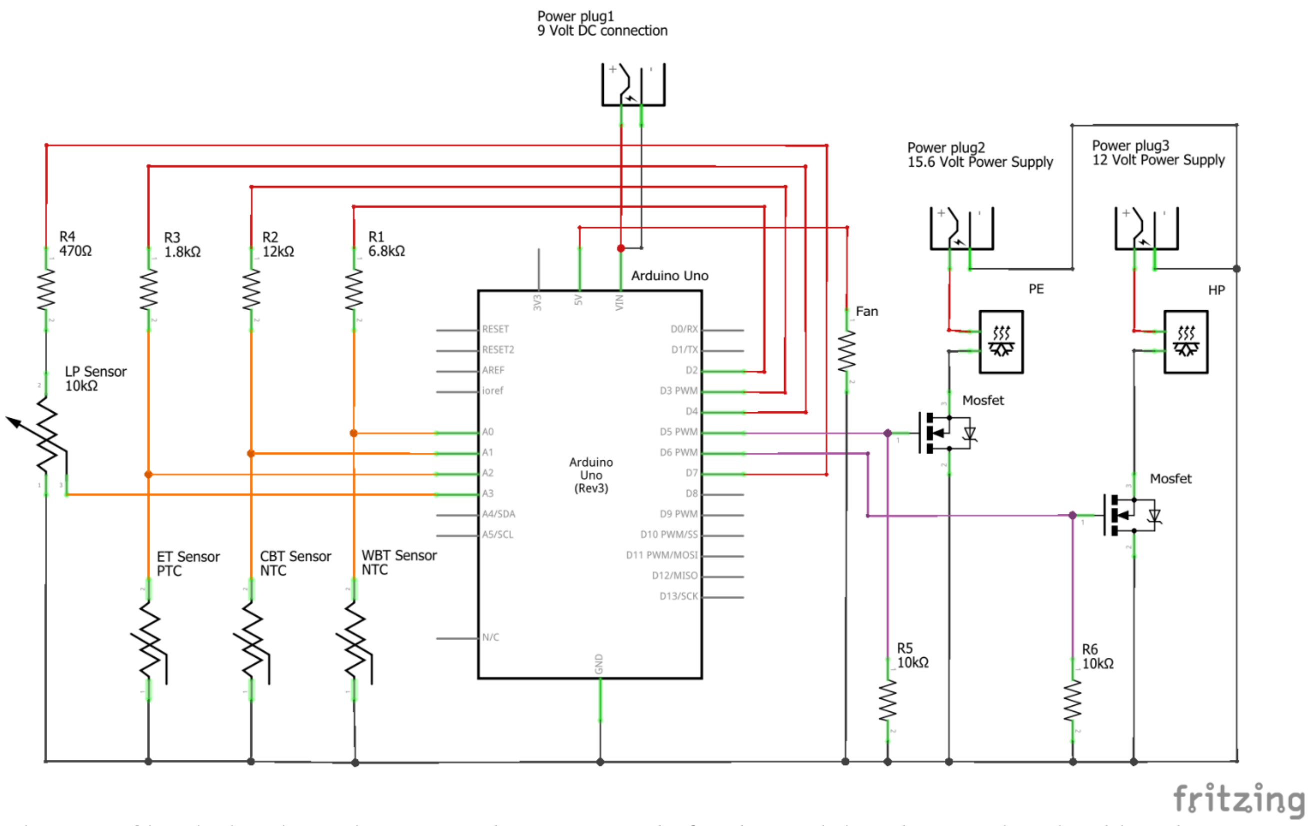


Figure 1: Circuit drawing using external power supply for the Arduino, heat pad and peltier element. In this drawing LP = Linear Potentiometer, ET = Environment Temperature, CBT = Cold Body Temperature, WBT = Warm Body Temperature, PE = Peltier element, HP = Heat pad

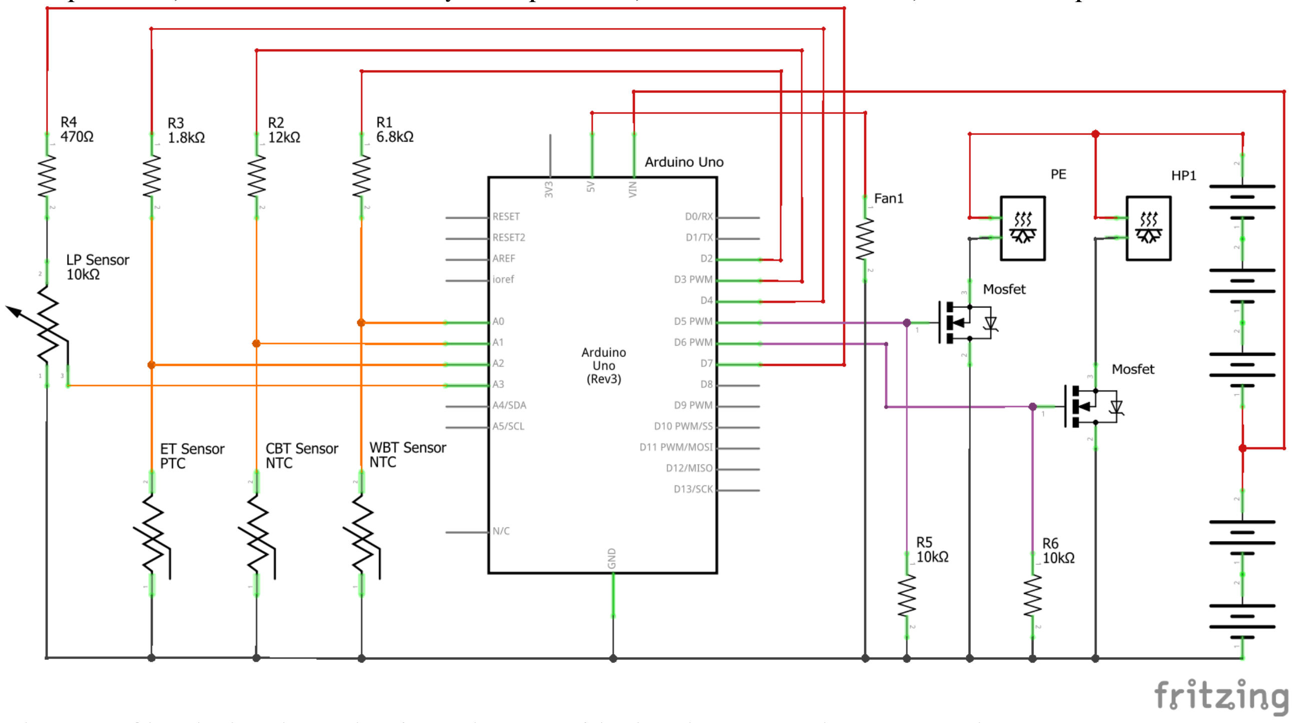


Figure 2: Circuit drawing using batteries. . In this drawing LP = Linear Potentiometer, ET = Environment Temperature, CBT = Cold Body Temperature, WBT = Warm Body Temperature, PE = Peltier element, HP = Heat pad

```
%% A file created to calculate the resistors giving the largest dV%%
clc
clear all
```

```
clear
close all

Vi = 5;         % V going in the system

%Tempresistors 5K Heatpad Peltier
Rvmaxt1 = 20000;  % Ohms, maximum resistance given by sensor -10 Celsius
Rvmint1 = 2000;   % Ohms, minimum resistance given by sensor 50 Celsius

RTemp1 = sqrt(Rvmaxt1*Rvmint1)  %Ohm

%Tempresistors 10K Heatpad Peltier
Rvmaxt2 = 40000;  % Ohms, maximum resistance given by sensor -10 Celsius
Rvmint2 = 4000;   % Ohms, minimum resistance given by sensor 50 Celsius

RTemp2 = sqrt(Rvmaxt2*Rvmint2)  %Ohm

%Tempsensor environment
Rvmaxt3 = 2417;   %Ohm, maximum resistance given by sensor
Rvmint3 = 1495;   %Ohm, minimum resistance given by sensor

RTemp3 = sqrt(Rvmaxt3*Rvmint3)  %Ohm

%Potresistors
Rvmaxp = 10660;  %Ohm, maximum resistance given by potmeter
Rvminp = 18;   %Ohm, minimum resistance given by potmeter

Rpot = sqrt(Rvmaxp*Rvminp)      %Ohm
```

Appendix 8: Arduino code

In the code for the Arduino first the pins are assigned, after which the variables are assigned and constants calculated. Then the void setup starts where the pins are turned on and after that comes the loop where everything else happens.
In these four parts of the code a constant order was maintained. This is done to keep a clear overview of the document. The order is as follows:

1. Warm body temperature sensor
2. Cold body temperature sensor
3. Environment temperature sensor
4. Potentiometer
5. Heating pad
6. Peltier element
7. Calculating the thermal conductivity
8. Printing the values on the screen.

For the temperature sensors the value read by the analog pins is an integer from 0 to 1023. This integer is converted in the voltage corresponding to that value, 0V corresponds with 0 and 5V corresponds with 1023. From there the resistance of the sensor at that time is calculated. The sensors need to be calibrated. The warm and cold body temperature sensors use the Steinhart-Hart equation [13]. The environment sensor uses an equation found in its document sheet. The environment sensor only needed to be calibrated at one temperature for the equation to work at different temperatures as well. This was done using a bucket of melting ice which is 0 degree Celsius. On basis of this calibrated sensor, the other temperature sensors are be calibrated as well. For the Steinhart-Hart equation 3 output resistances need to be found for 3 different temperatures. The temperatures used are a bucket of melting ice, 0 degrees Celsius, room temperature of 23.11 degrees Celsius and a bucket of heated water at 41 degrees Celsius. With these values the Steinhart constants were determined, which were implemented in the Arduino code. After the calibration the resistances of the sensors are transformed to temperature values in both Celsius and Kelvin. As stated in Appendix 7: 'Electrical circuit', the required ranges of the sensor outputs are spread out over the 1024 steps in order to maximize the sensitivity of the sensors.

For the potentiometer the output voltage is linear in relation to the distance travelled. Reading the value from the analog pin when the potentiometer is at the lowest position corresponds with 0 mm. Also the value when the potentiometer is at the highest position the value is read. Measuring the distance, with an (external) calibrated calliper, between the highest and lowest position a linear relation was created to calculate the distance travelled by the pin on the potentiometer.

In section 5 and 6 the heating pad and the peltier element are controlled by a PID control. The values are found experimentally until a stable system was achieved. For the heating pad the PID control values are kp = 8, ki = 3 and kd = 0.01. For the peltier element these values come to kp = 30, ki= 10 and kd = 5. For the heating pad the temperature difference between the warm body and the environment was the value to be controlled. The difference between those should be 10 degrees Celsius. The average of the last three values of the environment sensor where taken to calculate this to negate a sudden jump in the environment temperature. These jumps happen at points when the actual temperature is in between two values that can be read by the Arduino.

For the peltier element the difference between the cold body and the environment is being controlled. The desired difference between these two values should be 0 degrees Celsius. Here again the average of the last three environment temperature values is used.

After the PID value is determined, it is converted to a PWM (Pulse Width Modulation) signal the Arduino can use. First the PID value is converted into a integer ranging from 0 to 255. After that the PID value is checked against a hard cap value. If the PID value is higher than the hard cap value it will be set back to the hard cap value, to make sure the circuit doesn't overheat. The hard cap value is set at 120 out of 255, which means at a maximum the circuit is at maximum voltage for a period of

480µs per 1000µs. See Appendix 7: ‘Electrical circuit’ explanation on PWM. The hard cap value is the same for both the peltier element and the heat.

In section 7 the thermal conductivity of the sample is calculated. To do this the PWM value of the heating pad is used as from this value the power to keep the warm body 10 degrees warmer than the cold body can be calculated. This is calculated by a linear fit model. This model was created by notating both the voltage as the amperage for every PWM value.

After the generated power by the heat pad is calculated the power loss through insulation is being deducted. This gives the power going through the samples. Using formula (1) taken form Mills [7]:

(1) $$Q_{clothes} = \frac{\Delta T}{\frac{l_{sample}}{k_{sample}} * A_{alu} - 2 * \frac{l_{alu}}{k_{alu}} * A_{alu}}$$

Rearranging this formula gives formula (2) for the thermal conductivity k:

(2) $$k_{sample} = \frac{l_{sample}}{\Delta T * A_{alu} / Q_{clothes} - 2 * \frac{l_{alu}}{k_{alu}}}$$

Where k_{sample} is the thermal conductivity of the sample. l_{sample} is the thickness of the sample measured by the potentiometer. ΔT is the temperature difference between the warm body and the cold body measured by the appropriate sensors. A_{alu} is a the surface area of the aluminum bodies. $Q_{clothes}$ is the power going through the samples. I_{alu} is the thickness of the aluminum bodies and k_{alu} is the thermal conductivity of the aluminum bodies.

After the k_{sample} is calculated and the steady state has been reached, the calculated values are averaged. Steady state is reached after 435 seconds as calculated in Appendix 5: ‘Insulation’. After the measurement has run this amount of time the k values get added to each other and then divided by the amount of k values added, resulting in an average value. The longer the program runs after steady state has been reached, the more certain the thermal conductivity value becomes.

The last section of the code, section 8, is printing all values of interest on the monitor of the pc. There is an interval added here at 5000ms, which means the values only get displayed every 5 second to keep the screen readable.

Arduino Code

```
/* Troughout this document the following order has been used:
 *  1. Warm Body Temperature Sensor (WB)
 *  2. Cold Body Temperature Sensor (CB)
 *  3. Environment Temperature Sensor (E)
 *  4. Potentiometer (PM)
 *  5. Heating Pad  (HP)
 *  6. Peltier Element (PE)
 *  7. Calculating Thermal Conductiviy
 *  8. Printing (To read the values on screen)
 *  Everywhere. Which means this order is used when
 *  1. Defining variables
 *  2. Calculating constants
 *  3. Void Setup
 *  4. Void Loop


 */
```

```
// Defining Variables //
/* Pins */
int TempWarmBodyPinRead = A0;
int TempWarmBodyPinPower = 2;
int TempColdBodyPinRead = A1;
int TempColdBodyPinPower = 3;
int TempEnvironmentPin = A2;
int TempEnvironmentPower = 4;
int PotPin = A3;
int PotPower = 7;
int HeatPadPin = 6;
int PeltierElementPin = 5;

//'Calculating' Constants //
/*1. Warm Body Temperature Sensor(WB)*/
int WBRead;                                      // Stores value read by the arduino for the sensor; 0-1023
float TWB, TcWB, VWB, VoWB, RTWB;                 // Used Variables to calculate the Temperature of the Warm Body temperature sensor
float VinWB = 5;                                  // Voltage going in the sensor circuit
float CVoWB = VinWB/1023;                              // Voltage per step the arduino is able to read.
float RWB = 6800;                                  // Ohm, resistance in series to the temperature sensor
//WB Steinhart Constant constants
float AWB = 1.212613474*pow(10, -3);
float BWB = 2.33064152*pow(10, -4);
float CWB = 2.715745132*pow(10, -7);
float LRTWB, TKSWB, TKWB;                          // Used Variables to calculate the Temperature of the Warm Body temperature sensor
/*2. Cold Body Temperature Sensor (WB)*/
int CBRead;                                      // Stores value read by the arduino for the sensor; 0-1023
float TCB, TcCB, VCB, VoCB, RTCB;                 // Used Variables to calculate the Temperature of the Cold Body temperature sensor
float VinCB = 5;                                  // Voltage going in the sensor circuit
float CVoCB = VinCB/1023;                              // Voltage per step the arduino is able to read
float RCB = 12000;                                 // Ohm, resistance in series to the temperature sensor
//CB Steinhart Constant constants
float ACB = 1.179356240*pow(10, -3);
float BCB = 1.995722535*pow(10, -4);
float CCB = 4.351300849*pow(10, -7);
float LRTCB, TKSCB, TKCB;                          // Used Variables to calculate the Temperature of the Warm Body temperature sensor
/* 3. Environment Temperature Sensor (E)*/
int ERead;                                       // Stores value read by the arduino for the sensor; 0-1023
float RE = 1800;                                  // Ohm, resistance in series to the temperature sensor
float R25E = 2000;                                 // resistance of thermistor at 25 Celsius
float ktE, TcE, RTE, VoE, TKE;                         // Used Variables to calculate the Temperature of the Environment temperature sensor
float VinE = 5;                                  // Voltage going in the sensor circuit
float CVoE = VinE/1024;                              // Voltage per step the arduino is able to read
float TcEStart = 23.23;                            // Calibrated value added to formula according to datasheet
float aE = 7.88*pow(10, -3);                         // Value found in datasheet of sensor to calculate temperature
float BE = 1.937*pow(10, -5);                         // Value found in datasheet of sensor to calculate temperature
/* 4. Potentiometer (PM)*/
int PRead;                                       // Stores value read by the arduino for the sensor; 0-1023
float mmmaxPM = 59.8;                                // mm, value the potmeter should give at highest point of device
float mmminPM = 0;                                 // mm, value the potmeter should give at lowest point of device
float mmPM;                                       // Used Variables to calculate the distance traveled by the potentiometer
float mmStart = 28.26;                             // mm, value used to zero the potentiometer when device is at lowest point
float PMmax = 892;                                 // Value read by PotPin(A3) at heighest point of device
float PMmin = 291;                                 // Value read by PotPin(A3) at lowest point of device
float StepsPM = PMmax - PMmin;                         // Total steps measured by PotMeter in the range of device
float DStepsPM = mmmaxPM / StepsPM;                  // mm per step measured by the arduino
/* 5. Heating Pad  (HP)*/
float DTWEBHP = 10;                                // K, Desired Temperature Difference between Warm Body sensor and Environment sensor
float ADTWEBHP;                                    // Actual Temperature Difference between Warm Body sensor and Environment sensor
int PWMValueHP;                                    // PWM value output over HeatPadPin(11) used to control the heatpad (The PWMvalue*4microseconds is the width
                               //  of the pulse per millisecond and gets repeated every ms until the PWMvalue changes, PWMValue rangeg between 0 and 255)
float PWMValueHPi;                                 // intermediate PWM value
int HardCapValue = 120;                            // Maximum value PWM value is allowed to be, this is the same for the peltier element
// PID Constants
float kpWEB = 8.0;
float kiWEB = 3.0;
float kdWEB = 0.01;
float PID_EWEB, PIDHP, dStepHP, ETimeHP, TKE1, TKE2, TKE3;  // Variables used in the PID control for the Heatpad
unsigned long TimeHP;                              // s, Variable used to make timesteps for the PID control
unsigned long PreTimeHP = 0;                          // s, Variable used to make timesteps for the PID control
float PID_PEWEB = 0;                               // Setting start values for PID control on 0
float PID_pWEB = 0;
float PID_iWEB = 0;
float PID_dWEB = 0;
/* 6. Peltier Element (PE)*/
float DTCEBPE = 0;                                 // Desired Temperature Difference between Cold Body sensor and Environment sensor
float ADTCEBPE;                                    // Actual Temperature Difference between Cold Body sensor and Environment sensor
int PWMValuePE;                                    // PWM value output over HeatPadPin(11) used to control the heatpad
float PWMValuePEi;                                 // intermediate PWM value
// PID Constants
float kpCEB = 30.0;
float kiCEB = 10.0;
float kdCEB = 5.0;
float PID_ECEB, PIDPE, dStepPE, ETimePE;          // Variables used in the PID control for the Peltier Element
unsigned long TimePE;                              // s, Variable used to make timesteps for the PID control
unsigned long PreTimePE = 0;                          // s, Variable used to make timesteps for the PID control
float PID_PECEB = 0;                               // Setting start values for PID control on 0
float PID_pCEB = 0;
float PID_iCEB = 0;
float PID_dCEB = 0;
/* 7. Calculating Thermal Conductivity */
// Power
float QB0 = -0.042823;                             // Power output by heatpad is calculated by Q[W] = QB0 + QB1*PWMvalueHP
float QB1 = 0.12766;                               // Values found by linear fit through data found by PSU
float Qinsul = 0.5;                              // W, Calculated powerloss through the insulation
float QHP, Qclothes, QdeltaT;                      // Variables used in calculating the thermal conductivity of clothes
//Thermal
float lalu = 0.01;                                // m, thickness alu
float kalu = 147;                                 // W/mK, Thermal conducitivity alu
float Aalu = 0.06*0.06;                            // m2, Area aluminum
float Ralu = 2*(lalu/(kalu*Aalu));                 // K/W, Resistance Alu for Warm and Cold body combined
float Rtotal, kclothes, lclothes, kclothesi, kclothessum; //Variables used in calculating the thermal conductivity of clothes
unsigned long n = 0;                              // Total values of k calculated for average, changes in loop
unsigned long Counterk = 60000;                    // ms, time it takes for the system to reach steady state
unsigned long kTimer;                             // ms, used to see if steady state is reached
/* 8. Printing (To read the values on screen)*/
unsigned long Counter  = 0;                        // ms, keep track of when to show values on screen
const long interval = 5000;                       // ms, used to show value at intervals on screen


void setup() {
  Serial.begin(9600);
  // Power to Sensors
  pinMode(TempWarmBodyPinPower, OUTPUT);
  pinMode(TempColdBodyPinPower, OUTPUT);
  pinMode(TempEnvironmentPower, OUTPUT);
  pinMode(PotPower, OUTPUT);
  digitalWrite(TempWarmBodyPinPower, HIGH);
  digitalWrite(TempColdBodyPinPower, HIGH);
  digitalWrite(TempEnvironmentPower, HIGH);
  digitalWrite(PotPower, HIGH);
  //Heatpad output
  pinMode(HeatPadPin, OUTPUT);
  //Peltier output
  pinMode(PeltierElementPin, OUTPUT);
}

void loop() {


/* WarmBodyTemperatureSensor() */
  // Reading data from Warm Body Sensor and converting it //
  WBRead = analogRead(TempWarmBodyPinRead);       // Read Pin A0, gives value between 0-1023, depending on voltage over sensor
  VoWB = CVoWB*WBRead;                            // Convert read value to voltage
  RTWB = (VoWB*RWB)/(VinWB -VoWB);                // Calculate Resistance of Temp Sensor
  // Steinhart-hart equation broken down in easier blocks [1/T = A + B*ln(R) + C * (ln(R)^3)]
  LRTWB = log(RTWB);                              // Calculate ln(R)
  TKSWB = AWB + BWB*LRTWB + CWB*(pow(LRTWB, 3));  // 1/K, calculate 1/T
  TKWB = pow(TKSWB, -1);                          // K, calculate T in Kelvin
  TcWB = TKWB - 273;                              // C, calculate T in Celsius

/* ColdBodyTemperatureSensor() */
  // Reading data from Cold Body Sensor and converting it //
  CBRead = analogRead(TempColdBodyPinRead);       // Read Pin A1, gives value between 0-1023, depending on voltage over sensor
  VoCB = CVoCB*CBRead;                            // Convert read value to voltage
  RTCB = (VoCB*RCB)/(VinCB -VoCB);                // Calculate Resistance of Temp Sensor
  // Steinhart-hart equation broken down in easier blocks [1/T = A + B*ln(R) + C * (ln(R)^3)]
  LRTCB = log(RTCB);                              // Calculate ln(R)
  TKSCB = ACB + BCB*LRTCB + CCB*(pow(LRTCB, 3));  // 1/K, calculate 1/T
  TKCB = pow(TKSCB, -1);                          // K, calculate T in Kelvin
  TcCB = TKCB - 273;                              // C, calculate T in Celsius

/* EnvironmentTemperatureSensor() */
  //Reading data from Environment Sensor and converting it
  ERead = analogRead(TempEnvironmentPin);         // Read Pin A2, gives value between 0-1023, depending on voltage over sensor
  VoE = CVoE * ERead;                             // Convert read value to voltage
  RTE = (VoE*RE)/(VinE -VoE);                     // Calculate Resistance of Temp Sensor
  ktE = RTE/R25E;                                 // Calculate variable according to datasheet
  TcE = TcEStart + ((sqrt(pow(aE,2) - 4*BE + 4*BE*ktE)-aE)/(2*BE)); // C, Calculate T in C according to datasheet
  TKE = TcE + 273;                                // K, calculate T in Kelvin

/* PotentioMeter() */
  //Reading data from potentio meter and converting it
  PRead = analogRead(PotPin);                     // Read Pin A3, gives value between 0-1023, depending on voltage over sensor
  mmPM = DStepsPM * PRead - mmStart;              // mm, convert read value to mm

/* HeatingPad() */
   // Controlling the HeatPad
   TKE3 = TKE2;
   TKE2 = TKE1;
   TKE1 = TKE;
   TKE = (TKE1 + TKE2 + TKE3)/3;                  // K, Average Environment Temperature reading attenuate outliner
   ADTWEBHP = TKWB - TKE;                          // Calculating actual temperature difference between Warm body en environment
   PID_PEWEB = PID_EWEB;                          // Storing previous PID error value
   PID_EWEB = DTWEBHP - ADTWEBHP;                 // Calculating current PID error
   // Calculating the p of PID
   PID_pWEB = kpWEB*PID_EWEB;                     // Calculating p
   // Calculating the i of PID
   PreTimeHP = TimeHP;                            // ms, Storing previous time
   TimeHP = millis();                             // ms, Storing current time
   ETimeHP = (float)(TimeHP-PreTimeHP)/1000;      // s, calculatin time difference and converting to s
```

```
    PID_iWEB = PID_iWEB + (kiWEB * PID_EWEB)*ETimeHP; // calculating i
    // Calculating the d of PID
    dStepHP = PID_EWEB - PID_PEWEB;              // Calculating difference between previous error and current error
    PID_dWEB = kdWEB * ((dStepHP)/ETimeHP);       // Calculating d
    // PID
    PIDHP = PID_pWEB + PID_iWEB + PID_dWEB;       // Calculating PID = p+i+d
    // Changing the PID value to a PWM value
    PWMValueHPi = PIDHP;
    PWMValueHP = int(PWMValueHPi);
    if (PWMValueHP > HardCapValue){              // Creating a hard cap at 120 for PWM control to not melt the system
      PWMValueHP = HardCapValue;
    }
    if (PWMValueHP <=0){
      PWMValueHP = 0;
    }
    // Output

    analogWrite(HeatPadPin, PWMValueHP);         // Controlling the HeatPadPin(6) with the calculated PWM value

/*PeltierElement() */
    // Controlling the Peltier
    ADTCEBPE = TKE - TKCB;                       // Calculating actual temperature difference between Cold body en environment
    PID_PECEB = PID_ECEB;                        // Storing previous PID error value
    PID_ECEB = DTCEBPE - ADTCEBPE;               // Calculating current PID error
    // Calculating the p of PID
    PID_pCEB = kpCEB*PID_ECEB;                   // Calculating p
    // Calculating the i of PID
    PreTimePE = TimePE;                          // ms, Storing previous time
    TimePE = millis();                           // ms, Storing current time
    ETimePE = (float)(TimePE-PreTimePE)/1000;    // s, calculatin time difference and converting to s
    PID_iCEB = PID_iCEB + (kiCEB * PID_ECEB)*ETimePE; // calculating i
    // Calculating the d of PID
    dStepPE = PID_ECEB - PID_PECEB;              // Calculating difference between previous error and current error
    PID_dCEB = kdCEB * ((dStepPE)/ETimePE);       // Calculating d
    //PID
    PIDPE = PID_pCEB + PID_iCEB + PID_dCEB;       // Calculating PID = p+i+d
    // Changing the PID value to a PWM value
    PWMValuePEi = PIDPE;
    PWMValuePE = int(PWMValuePEi);               // Creating a hard cap at 120 for PWM control to not melt the system
    if (PWMValuePE > HardCapValue){
      PWMValuePE = HardCapValue;
    }
    if (PWMValuePE <=0){
      PWMValuePE = 0;
    }
    // Output
    analogWrite(PeltierElementPin, PWMValuePE);   // Controlling the PeltierElementPin(5) with the calculated PWM value

/*  Calculating Thermal Conductivity */
    // Calculating power
    QHP = QB0 + QB1*PWMValueHP;                  // W, Calculating the power the Heatpad uses to keep the warm body at the desired tempterature
    Qclothes = QHP - Qinsul;                     // W, Taking away the power lost through the insulation
    if (Qclothes < 0){                           // Keeping it positive
      Qclothes = 0;
    }
    QdeltaT = TKWB - TKCB;                       // K, Temperature difference between Warm Body and Cold Body
    // Calculating Thermal values
    lclothes =  mmPM/1000;                       // m, thickness of sample
    kclothesi = lclothes/((QdeltaT*Aalu/Qclothes)-(2*lalu/kalu));  // W/mK, thermal conductivity of clothes, intermediate value
    kTimer = millis();                           // ms, steady state timer
    if (kTimer >= Counterk){                     // This statement happens after steady state has been reached
      n = n + 1;                                 // Counter how often this loop has been done
      kclothessum = kclothesi + kclothessum;      // Add intermediate values together
      kclothes = (kclothessum)/n;                 // W/mK, average, divide sum of intermediate values after steady state by amount of calculated intermediate values after steady state
has been reached
    }

/* Print */
  // Print values on screen after a set interval
  unsigned long PrintTime = millis();
  if (PrintTime - Counter >= interval){
    Counter = PrintTime;
    // Print Warm body
    Serial.print("Warm Body: ");
//    Serial.print(WBRead);
//    Serial.print(" Read ");
//    Serial.print(VoWB);
//    Serial.print(" Volt ");
    Serial.print(TcWB);
    Serial.println(" C ");
    // Print Cold body
    Serial.print("Cold Body: ");
//    Serial.print(CBRead);
//    Serial.print(" Read ");
//    Serial.print(VoCB);
//    Serial.print(" Volt ");
//    Serial.print(RTCB);
//    Serial.print(" Ohm ");
    Serial.print(TcCB);
    Serial.println(" C ");
    // Print Environment
    Serial.print("Environment: ");
//    Serial.print(ERead);
//    Serial.print(" Read ");
//    Serial.print(VoE);
//    Serial.print(" Volt ");
    Serial.print(TcE);
    Serial.println(" C ");
    //Print Potmeter
    Serial.print("Potmeter: ");
    Serial.print(mmPM);
    Serial.println(" mm");
    //Print HeatPad
    Serial.print("Heatpad: ");
    Serial.print(PWMValueHP);
    Serial.println(" PWMHP ");
    //Print Peltier
    Serial.print("Peltier: ");
    Serial.print(PWMValuePE);
    Serial.println(" PWMPE ");
    Serial.print("Thermal conductivity: ");
    Serial.print(kclothesi);
    Serial.print(" Intermediate ");
    Serial.print(kclothes, 6);
    Serial.println(" Steady State ");
    Serial.print(kclothessum);
    Serial.print(" k Sum ");
    Serial.print(kTimer);
    Serial.println(" Time ");
    Serial.print(PID_dWEB);
    Serial.print(" d ");
    Serial.print(PID_iWEB);
    Serial.print(" i ");
    Serial.print(PID_pWEB);
    Serial.println(" p ");
    Serial.println("");
    }
}
```

Appendix 9: Thermal conductivity measurement uncertainty

Defining Aspired Precision

As stated in the report, a measurement accuracy within 8% of the real thermal conductivity, is desired. This appendix grants insight in the aspects contributing to the accuracy of the Therminus device. Formula (1), which efficaciously describes the thermal conductivity, is derived for the Therminus device specifically.

(1) $$k_{textile} = d_{textile}\left(\frac{A*\Delta T}{Q} - 2*\frac{d_{element}}{k_{element}}\right)^{(-1)}$$

Where A is the surface area (in m^2) on which the measurement takes place. ΔT is the temperature difference (in K) between the warm and cold body. ΔT is a value for which engineers ought to choose an appropriate value. Q is the heat flow (in Watt) needed to sustain the aforementioned temperature difference ΔT.

Parameter Remarks

Some parameters either have no, or do not have known, uncertainties. The parameter with unknown uncertainty is: $k_{element}$. Geometric uncertainties can be accounted for by measuring the real value after the manufacturing process has been completed. Examples of such values are: A and $d_{element}$. The uncertainty of Q is caused two factors $Q = Q_{supplied} - Q_{insulation}$. First factor ($Q_{supplied}$) being the uncertainty of the voltage drop, and the resistance the heat pad has, in practice. In other words, the first factor is caused by the uncertainty associated with the electrical circuit design. The second factor ($Q_{insulation}$) is caused by the error that is made by estimating the heat loss to the insulation. The prototype Therminus device does not suffer from the latter, it rather suffers from an error made by the former instead.
Consequently, three uncertainties remain. Both ΔT and $d_{textile}$ are caused in part by sensor inaccuracy and the magnifying effect caused by the discrete voltage mapping as carried out by Arduino. To establish the maximum uncertainty corresponding to both ΔT and $d_{textile}$, MATLAB files have been created to simulate the Arduino and, consequently, the error caused by the setups. Appendix 10: 'Thermistor uncertainty' (for $\sigma_{\Delta T}$) and Appendix 11: 'Linear potentiometer uncertainty' (for $\sigma_{d_{textile}}$) contain more information about the origin of the uncertainties alongside the aforementioned MATLAB files.
The uncertainty of Q remains, which could be regarded as the most complex one to analyze. In part because the formulae used to derive Q are quite nasty. Never mind the fact that the formulae used are an attempt to describe reality, placing emphasis on the word attempt. In other words, the exact uncertainty for Q, along with its origins, remain unsure. Although no conclusive answer can be given as to what value σ_Q will turn out to be in reality, an estimate for its maximum acceptable value will be established.

General Formula Error Propagation

The uncertainty of the thermal conductivity measurement is caused by the propagation of two errors. The first error is caused by the inaccurate measurement of ΔT. The second error is caused by the inaccuracy of the thickness measurement. Formula (2) defines how error propagation takes place.

(2) $$\sigma_{k_{textile}} = \sqrt{\left(\frac{\partial k_{textile}}{\partial \Delta T}\right)^2 (\sigma_{\Delta T})^2 + \left(\frac{\partial k_{textile}}{\partial d_{textile}}\right)^2 (\sigma_{d_{textile}})^2 + \left(\frac{\partial k_{textile}}{\partial Q}\right)^2 (\sigma_Q)^2}$$

To determine all components in formula (2), the members ($\frac{\partial k_{textile}}{\partial \Delta T}$ and $\frac{\partial k_{textile}}{\partial d_{textile}}$) are derived independently. Note that both $\sigma_{\Delta T}$ and $\sigma_{d_{textile}}$ are results attained from theoretical analysis which are analyzed in their respective appendices.

Formulae $\boldsymbol{\Delta T}$ *Error Propagation*

From formulae (3), (4) and (5), formula (6) can be derived. Note that formula (6) merely formulates a result attained by applying u-substitution as defined by formulae (3) through (5).

(3) $$k_{textile} = \frac{d_{textile}}{u}$$

(4) $$u = \frac{A}{Q}\Delta T - 2\frac{d_{element}}{k_{element}}$$

(5) $$\frac{\partial k_{textile}}{\partial \Delta T} = \frac{\partial k_{textile}}{\partial u}\frac{\partial u}{\partial \Delta T}$$

(6) $$\boxed{\frac{\partial k_{textile}}{\partial \Delta T} = -\frac{d_{textile}*A*\Delta T}{Q}\left(\frac{A*\Delta T}{Q} - 2*\frac{d_{element}}{k_{element}}\right)^{(-2)}}$$

Now that formula (6) has been defined, it can be substituted back into formula (2).

Formulae $\boldsymbol{d_{textile}}$ *Error Propagation*

The derivation $\frac{\partial k_{textile}}{\partial d_{textile}}$ of is much more straightforward than the derivation examined in the former paragraph. Formula (7) depicts the relationship of $k_{textile}$ derived with respect to $d_{textile}$.

(7) $$\boxed{\frac{\partial k_{textile}}{\partial d_{textile}} = \left(\frac{A*\Delta T}{Q} - 2*\frac{d_{element}}{k_{element}}\right)^{(-1)}}$$

Formulae $\boldsymbol{Q}$ *Error Propagation*

Before discussing the derivation of $k_{textile}$ with respect to Q, an important remark has to be made. The assertion that equation (8) can be used, infers that the formula used to describe $Q_{insulation}$ is exactly right and introduces no additional error. During the analysis, the assumption that $Q_{insulation}$ is indeed exactly right, is made. A more in depth analysis into the derivation of $Q_{insulation}$ is given in Appendix 5: 'Insulation'.

(8) $$Q = Q_{supplied} - Q_{insulation}$$

(9) $$\frac{\partial Q}{\partial Q_{insulation}} = -1$$

(10) $$\frac{\partial k_{textile}}{\partial Q_{insulation}} = \frac{\partial k_{textile}}{\partial Q}\frac{\partial Q}{\partial Q_{insulation}}$$

(11) $$\frac{\partial k_{textile}}{\partial Q} = d_{textile} * A * \Delta T * \left(Q * \left(\frac{A*\Delta T}{Q} - 2*\frac{d_{element}}{k_{element}}\right)\right)^{(-2)}$$

$$(12) \quad \frac{\partial k_{textile}}{\partial Q_{insulation}} = d_{textile} * A * \Delta T * ((Q_{supplied} - Q_{insulation}) * \left(\frac{A*\Delta T}{Q_{supplied} - Q_{insulation}} - 2 * \frac{d_{element}}{k_{element}} \right))^{(-2)}$$

Resulting Uncertainty Formula

A MATLAB script has been used to solve equation (2). Equation (13) can be constructed from the set of derived formulae (6), (7) and (12).

$$(13) \quad {\sigma_{k_{textile}}}^2 = (\frac{d_{textile}*A*\Delta T}{Q}(\frac{A*\Delta T}{Q} - 2 * \frac{d_{element}}{k_{element}})^{(-2)})^2 (\sigma_{\Delta T})^2 + ((\frac{A*\Delta T}{Q} - 2 * \frac{d_{element}}{k_{element}})^{(-1)})^2 (\sigma_{d_{textile}})^2 - d_{textile} * A * \Delta T * ((Q_{supplied} - Q_{insulation}) * \left(\frac{A*\Delta T}{Q_{supplied} - Q_{insulation}} - 2 * \frac{d_{element}}{k_{element}} \right))^{(-2)} (\sigma_{Q_{insulation}})^2$$

The error $\sigma_{k_{textile}}$ ought not to exceed 8% of the actual thermal conductivity, see Appendix 2: 'Phoebe experiment'. One can obviously deduct that the narrowest bounds are applicable when $k_{textile}$ is as small as possible.
As previously stated in the smallest thermal conductivity which is expected to be the worst case $k_{textile} = 0.057 \frac{W}{mK}$. Thus logically leading to an allowable error of $\pm 4.56 * 10^{(-3)}$. Two cases are evaluated in order to establish the prototype performance.

Two Evaluated Cases

First case being the most extreme case being a well insulating thick coat in an environment temperature that does not exceed 30 C°. The Arduino Uno is deemed inappropriate due to several reasons. The first case for which the maximum uncertainty, with its respective conditions, assumes an Arduino Uno is implemented. The second case which is analyzed, assumes an SAM3X controller precision is implemented to read sensor values for equivalent circuitry.
For both cases the error is evaluated for the most critical case. The most critical case assumes a thermal resistivity of R_{th} = 2.0826 where $R_{th} = \frac{k}{d}$ = 0.057/0.02737. As mentioned in Appendix 6: 'Heat pad', 95.45% of coats is expected to be thinner than 27.37mm.

Uncertainty Values

Tables 1 and 2 list the worst case uncertainties that are expected for the Arduino Uno and Due respectively. Note that the uncertainties are calculated from several MATLAB files.
Table 1: Highest expected uncertainty for respective quantity using the Arduino Uno

Case 1		Arduino Uno
Quantity	Value unit	At condition
$\sigma_{\Delta T}$	6.3896e-02 C°	At Tenvironment = 39.34 C°
$\sigma_{d_{textile}}$	4.869e03 mm	At dtextile = 0.4 mm
σ_Q	1.7694e-05 W	At Tenvironment = 11.48 C°

Table 2: Highest expected uncertainty for respective quantity using the Arduino Due

Case 2		Arduino Due
Quantity	Value unit	At condition
$\sigma_{\Delta T}$	2*1.6417e-02 C°	At Tenvironment = 38.85 C°
$\sigma_{d_{textile}}$	1.221e-03mm	At dtextile = 23.56 mm
σ_Q	1.7694e-05 W	At Tenvironment = 11.48 C°

Note that for both Arduino Uno and Arduino Due the value for σ_Q are equal since the same interpolation method is used in order to calculate the heat loss through the insulation.

Resulting Error Estimate

Tables 3 and 4 list the results that are attained from MATLAB script Therminus_Error_Propagation which is provided at the end of this Appendix. Table 3 provides the expected values as attained from the aforementioned MATLAB script when an Arduino Uno is used.

Table 3: highest expected uncertainty for respective quantity using the Arduino Uno

Case 3		Arduino Uno
Quantity	Value unit	Relative Uncertainty
$\frac{\partial k_{textile}}{\partial \Delta T}$	-0.05702	0.639%
$\frac{\partial k_{textile}}{\partial d_{textile}}$	2.036	0.0178%
$\frac{\partial k_{textile}}{\partial Q_{insulation}}$	-0.7782	0.0227%
$\sigma_{k_{textile}}$	0.0123 W/mK	21.58%

Table 4: highest expected uncertainty for respective quantity using the Arduino Uno

Case 4		Arduino Due
Quantity	Value unit	Relative Uncertainty
$\frac{\partial k_{textile}}{\partial \Delta T}$	-0.05702	0.1642%
$\frac{\partial k_{textile}}{\partial d_{textile}}$	2.036	0.0045%
$\frac{\partial k_{textile}}{\partial Q_{insulation}}$	-0.7782	0.0227%
$\sigma_{k_{textile}}$	0.0028 W/mK	4.9528%

Lineair Interpolation Heat Loss Model Justification

If the heat flow through the insulation would be determined accurately instead of using lineair interpolation, both resulting errors for case 3 and 4, will become smaller.
For the Arduino Uno the reduced error is of no important matter since the error associated with the still exceeds the desired 8%. When perfectly implementing the insulation heat loss model, the expected error for the Arduino Uno is 20.29%, this is an improvement. While theoretically possible to implement such model, it will be extremely cumbersome in practice and therefore not recommended.
For the Arduino Due the reduction in the total error made is less significant, namely 4.9527%. A more interesting matter is the maximum error that can be made for the insulation heat loss estimate. For the Arduino Due, the aforementioned maximum error is 7.688%, or in an absolute sense, 5.993e-03. The error made using the proposed linear interpolation is approximately 340 times smaller than the maximum allowable error thus justifying the usage of linear interpolation.

```
% Therminus Error Propagation
% TU Delft, 05/31/2019
% Author: Sebastiaan Thomas Njio

clc
close all
clear variables

%Important values
dtextile = 28*10^(-3);
k_accuracy = 1 - 0.08;            % 1 - Accuracy
Qmeasurement = [0.0732654, 0.0732654]; % [Case 1; Case 2] (Both the same value)
```

```
Qinsulationreal = [0.0783692, 0.0774926]; % [Case 1; Case 2], This one is case specific

%Qinsulation
Qinsulation = [0.077961, 0.0774926]; %[Lin Interpolation max Error @ 11.48C]

% Values attained for 12 Bit arduino measurement with Calibrated Sensors
sdT = [2*1.6417e-02, 2*1.2970e-02];     %[@T = 38.85, @T = -9.138]
sdtextile = [1.038e-03, 1.038e-03];      %[@d = 27.98mm, @d = 27.98mm]



% Values for 10 Bit arduino measurement with Calibrated Sensors
%sdT = [2*6.3896e-02, 2*5.1713e-02];     %[@T = 39.34, @T = -9.879]
%sdtextile = [4.869e-03, 0.004886];       %[@d = 0.4, @d = 71.52mm]
%sdtextile = [4.869e03 , 4.767e-03];       %[@d = 0.4mm, @d = 28.02mm]

sQ = [1.7694e-05, 1.7694e-05];            %[@T = -10, @T= -10]

%Constants
delement = 10*10^(-3);
kelement = 147;
b = 60*10^(-3);
l = 60*10^(-3);
dT = 10;
Q = zeros(1,length(Qinsulation));

%Values which need to be calculated
sktextile = zeros(2,length(Qinsulation));
skmax = zeros(1,length(Qinsulation));
ddT = zeros(1,length(Qinsulation));
ddtextile = zeros(1,length(Qinsulation));
ktextile = zeros(1,length(Qinsulation));
dQ = zeros(1,length(Qinsulation));

A = b*l;
for i = 1:length(Qinsulation)
Q(1,i) = Qinsulation(1,i)+Qmeasurement(1,i);
   ddT(1,i) = - dtextile*A*dT/Qmeasurement(1,i)  *  (A*dT/Qmeasurement(1,i) -2*delement/kelement)^(-2);
   ddtextile(1,i) = 1/(A*dT/Qmeasurement(1,i) - 2*delement/kelement);
   dQ(1,i) = -dtextile*A*dT*((Qmeasurement(1,i)*(A*dT/Qmeasurement(1,i) - 2*delement/kelement))^(-2));

   sktextile(1,i) = sqrt((ddT(1,i)*sdT(1,i))^2 + (ddtextile(1,i)*sdtextile(1,i))^2);
   sktextile(2,i) = sqrt((ddT(1,i)*sdT(1,i))^2 + (ddtextile(1,i)*sdtextile(1,i))^2 + (dQ(1,i)*sQ(1,i))^2);

   ktextile(1,i) = dtextile/(A*dT/Qmeasurement(1,i) - 2*delement/kelement);

   % Determine max allowed sQ in order to remain at k_accuracy
   skmax(1,i) = (1-k_accuracy)*ktextile(1,i);
   sQ(1,i) = sqrt(((skmax(1,i))^2  -  (ddT(1,i)*sdT(1,i))^2 + (ddtextile(1,i)*sdtextile(1,i))^2)*(dQ(1,i))^(-2));



end

%OUTPUTS to user
disp(['dktextile/ddT = ', num2str(ddT(1,1),4)]);
   disp(['dktextile/ddtextile = ', num2str(ddtextile(1,1),4)]);
   disp(['dktextile/dQ = ', num2str(dQ(1,1),4)]);
   disp(['Error without Qerror: ', num2str(sktextile(1,1)/ktextile(1,1)*100, 8), '% with k = ', num2str(ktextile(1,1),4)])
   disp(['Allowable Qmeasurement Error [Absolute, Relative]: [', num2str(sQ(1,1), 4), ', ',
num2str(sQ(1,1)/Qinsulation(1,1)*100,4),'%]'])
    fprintf('\n'); disp(['Relative error [No Q error]: ',num2str(100*sktextile(1,1)/ktextile(1,1)),'%'])
    disp(['Relative error [Error, @Qerror %]: [',num2str(100*sktextile(2,1)/ktextile(1,1)),'%, ',num2str((Qinsulation(1,1)-
Qinsulationreal(1,1))/Qinsulationreal(1,1)*100),'%]']);fprintf('\n')
```

Appendix 10: Thermistor uncertainty

This Appendix aims to provide insight into uncertainty that is caused by the sensor setup.

Thermistor Sensor Setup

The setup and its resistor values (in order to maximize the voltage range) is described in Appendix 7: ‘Electrical Circuit’. The values that are given in this Appendix are calculated on the basis of a 12bit logic board. Note that the Arduino used in the current Therminus device might not have 12bit read capability on analog read ports. Hence an Arduino with 12bit read capabilities is recommended.

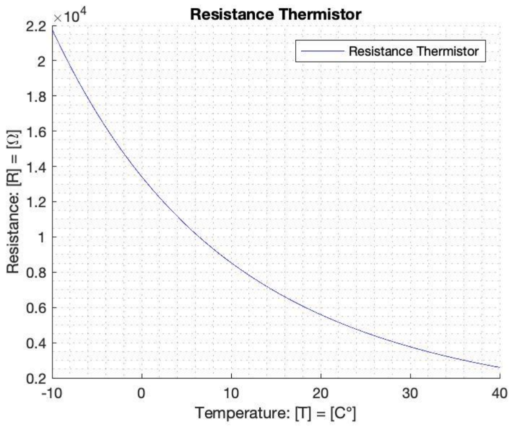


Figure 1: Calibrated thermistor resistance as function of temperature

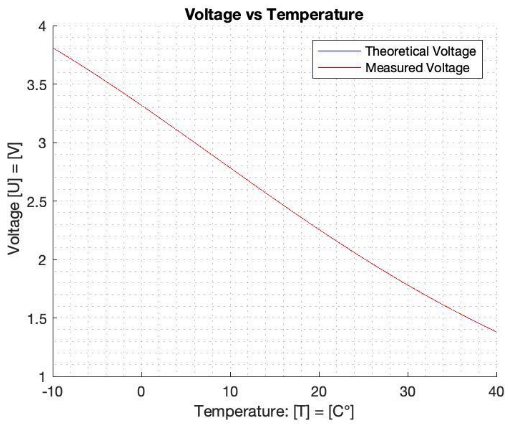


Figure 2: Measured voltage at temperature

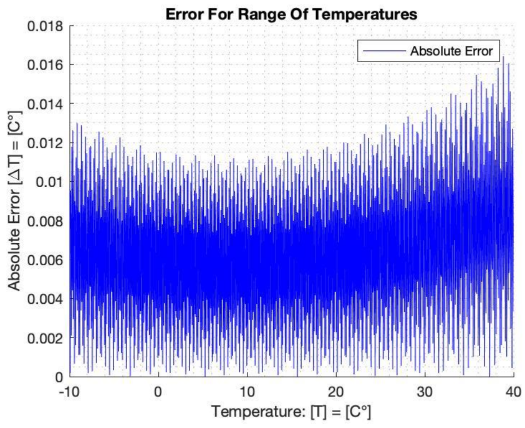


Figure 3: The absolute error resulting from the rounding error caused by discrete Arduino values

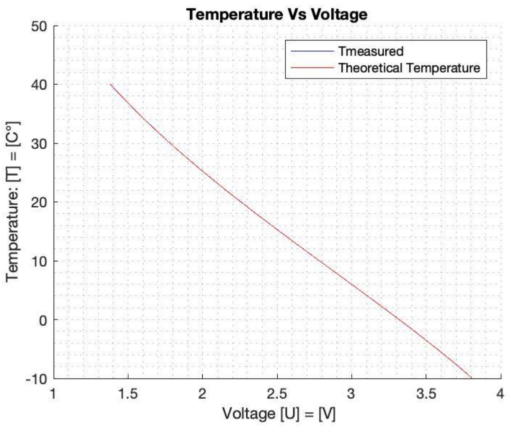
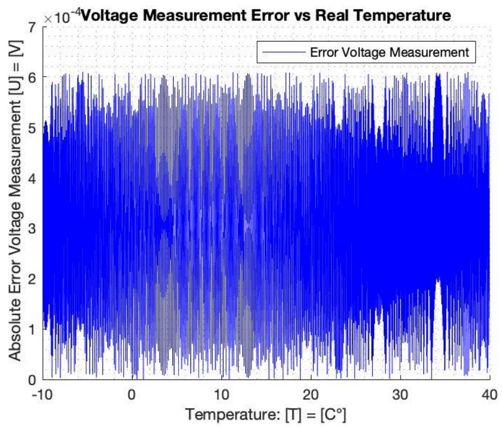


Figure 4: Fractional power loss for range of expected T_∞ values



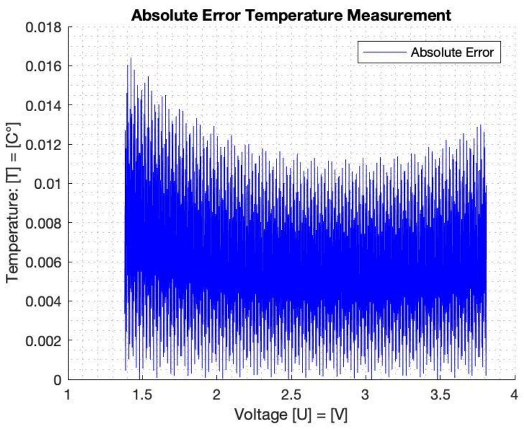


Figure 5: Absolute voltage error expected to occur at certain temperature

Figure 6: Absolute temperature error at measured voltage

Important values from the script are extracted and displayed to the user. The parameters are given at the end of the Appendix.

Parameters as Used in Error Propagation Appendix

To determine the maximum temperature delta error $\sigma_{\Delta T}$. The temperature difference is measured between two sensors, both sensors are assumed to cause the same (maximum) error 1.6417e-02. The maximum absolute error is caused by the combination of the discrete voltage measurement (by Arduino) the required voltage read to resistance mapping, and the resistance to temperature mapping. Please note that errors caused by floating point approximation is not considered.
As previously mentioned, two sensors are used thus resulting in a maximum error $\sigma_{\Delta T} = 2 * (1.6417\mathrm{e} - 02)$.

MATLAB Script Outputs

```
Steinhart-Hart Equation constants
AR: 1.0157e-03 | BR: 2.7418e-04 | CR: 4.5791e-08

Temperature Delta Body: 10 C°
Measured Voltage [max, min]: [3.8106, 1.3811]
Uncertainty Voltage Measurement Arduino: 1.2210e-03 Volt

Expected Measured Arduino Step Range [max, min]: [3121, 1131]
Usefull occupancy of arduino voltage range: 48.6%
Max Temperature Uncertainty Sensor [error, @T]: [1.6417e-02 C°, 38.85 C°]
Maximum Relative Uncertainty: 0.1642%

Max Temperature Error for T < 8 C°, [Error, @T] : [1.2970e-02 C°, -9.138 C°]
Maximum Relative Uncertainty: 0.1297%
```

```
% Temperature Sensor Calibrated
% TU Delft, 05/31/2019
% Author: Sebastiaan Thomas Njio

clc
clear all
close all

Resolution = 2^12-1; % Resolution of Arduino Volt measurement
T = 263.15:0.02:313.15; % Temperature range which is expected
Rchosen = 6.8*10^3;
Tplot = T -273.15;

%Measured Steinhart-Hart Equation Values (1/T = A + B*ln(R) + C(ln(R)^3)
AR = 1.015659556e-3; BR = 2.741829352e-4; CR = 0.4579094890e-7;

%Allocate RAM for variables
Rthermistor = zeros(1,length(T));
Utheoretical = zeros(1,length(Rthermistor));
Umeasured = zeros(1,length(Rthermistor));
```

```
reg = zeros(4,length(T));

for i = 1:length(T)
x = 1/(2*CR)*(AR -1/T(i)); y = sqrt((BR/(3*CR))^3 + x^2);

Rthermistor(1,i) = exp((y-x)^(1/3)-(y+x)^(1/3));
Utheoretical(1,i) = 5 * Rthermistor(1,i) / (Rchosen + Rthermistor(1,i));
Umeasured(1,i) = 5/Resolution*round(Utheoretical(1,i)*Resolution/5);

end

%Important Vectors and Variables
Rthmax = max(Rthermistor); Rthmin = min(Rthermistor);
Umax = max(Utheoretical(1,:)); Umin = min(Utheoretical(1,:));

Ux = linspace(0, 5, Resolution +1);
Uxp = Umin:0.002:Umax;
Ux = 5/Resolution*round(Uxp*Resolution/5);

Rmeasured = zeros(1,length(Ux));
Ttheoretical = zeros(1,length(Uxp));
Tmeasured = zeros(1,length(Ux));
reg2 = zeros(2,length(Ux));
for i = 1:length(Uxp)
   Ttheoretical(1,i) = 1/(AR + BR*log(Rchosen*Uxp(1,i)/(5-Uxp(1,i))) + CR*(log(Rchosen*Uxp(1,i)/(5-Uxp(1,i))))^3) -
273.15;
   Tmeasured(1,i) = 1/(AR + BR*log(Rchosen*Ux(1,i)/(5-Ux(1,i))) + CR*(log(Rchosen*Ux(1,i)/(5-Ux(1,i))))^3) - 273.15;
   if i<length(Uxp)
   reg2(1,i) = Ux(1,i+1) - Ux(1,i); %Log the Voltage difference made between steps
   reg2(2,i) = Uxp(1,i);
   end
end

%Important Paramaters
smin = floor(Umin*Resolution/5); smax = ceil(Umax*Resolution/5);
Terror = abs(Tmeasured-Ttheoretical);
Uerror = abs(Umeasured - Utheoretical);

[Termax, Terrmax_i] = max(Terror);

%% Plotting Figures
disp(['*Note that values will be reported in rounded form.']);
disp(['For accurate representation consult the script.']); fprintf('\n')
disp(['Steinhart-Hart Equation constants'])
disp(['AR: ',num2str(AR,'%.4e'),' | ', 'BR: ',num2str(BR,'%.4e'),' | ', 'CR: ',num2str(CR,'%.4e'),]); fprintf('\n')
disp(['Temperature Delta Body: 10 C',char(176)])
disp(['Measured Voltage [max, min]: [',num2str(Umax),', ', num2str(Umin),']'])
disp(['Uncertainty Voltage Measurement Arduino: ',num2str(5/Resolution,'%.4e'),' Volt']); fprintf('\n')
disp(['Expected Measured Arduino Step Range [max, min]: ','[',num2str(smax),', ', num2str(smin),']'])
disp(['Usefull occupancy of arduino voltage range: ',num2str((smax-smin)/Resolution*100,4),'%'])
disp(['Max Temperature Uncertainty Sensor [error, @T]: [', num2str(Terrmax,'%.4e'),' CÔøΩ,
',num2str(Ttheoretical(1,Terrmax_i),4),' C',char(176),']']);
disp(['Maximum Relative Uncertainty: ',num2str(Terrmax*10,4),'%']);fprintf('\n')

%Establishing max Error made untill 8C
i = max(find(fliplr(Ttheoretical) <= 8));
[Terror8, index] = max(Terror(1,i:end));
index = i-1+index;
```

```
disp(['Max Temperature Error for T < 8 CÔøΩ, [Error, @T] : [', num2str(Terror8,'%.4e'),' CÔøΩ,
',num2str(Ttheoretical(1,index),4),' CÔøΩ]'])
disp(['Maximum Relative Uncertainty: ',num2str(Terror8*10,4),'%']);fprintf('\n')

%% Figure Drawing
%                               ///       FIRST FIGURES        \\\
figure()                % PLOT HIGH FIDELITY VOLTAGE VS TEMPERATURE
hold on, grid minor;
x0 = 100; y0 = 600; height = 300; width = 400;
set(gcf,'units','points','position',[x0,y0,width,height]);
title('Resistance Thermistor');xlabel(['Temperature: [T] = [C',char(176),']']); ylabel('Resistance: [R] = [\Omega]');
plot(Tplot, Rthermistor,'-b')
legend('Resistance Thermistor','Location','northeast')

figure()               % Error
hold on, grid minor;
x0 = 100; y0 = 600 - (80+height);
set(gcf,'units','points','position',[x0,y0,width,height]);
title('Error For Range Of Temperatures');xlabel(['Temperature: [T] = [C',char(176),']']); ylabel(['Absolute Error [\DeltaT] =
[C',char(176),']']);
plot(Ttheoretical, Terror,'-b')
legend('Absolute Error','Location','northeast')

%                               ///       SECOND FIGURES       \\\

figure()               % PLOT REAL TEMP, MEASURED TEMP FOR MEASURED VOLTAGE
hold on, grid minor;
x0 = 100 + (20+width); y0 = 600;
set(gcf,'units','points','position',[x0,y0,width,height]);
title('Voltage vs Temperature'); xlabel(['Temperature: [T] = [C',char(176),']']); ylabel([' Voltage [U] = [V]']);
plot(Tplot, Utheoretical,'-b', Tplot, Umeasured, '-r')
legend('Theoretical Voltage','Measured Voltage','Location','northeast')

figure()               % PLOT ERROR VOLTAGE VS TEMPERATURE
hold on, grid minor;
x0 = 100 + 1*(20+width); y0 = 600 - 1*(80+ height);
set(gcf,'units','points','position',[x0,y0,width,height]);
title('Voltage Measurement Error vs Real Temperature');xlabel(['Temperature: [T] = [C',char(176),']']); ylabel(' Absolute
Error Voltage Measurement [U] = [V]');
plot(Tplot,Uerror,'-b')
legend('Error Voltage Measurement','Location','northeast')

%                               ///       TERTIARY FIGURES      \\\

figure()               % PLOT REAL TEMP, MEASURED TEMP FOR MEASURED VOLTAGE
hold on, grid minor;
x0 = 100 + 2*(20+width); y0 = 600;
set(gcf,'units','points','position',[x0,y0,width,height]);
title('Temperature Vs Voltage'); xlabel([' Voltage [U] = [V]']); ylabel(['Temperature: [T] = [C',char(176),']']);
plot(Ux,Ttheoretical,'-b',Ux,Tmeasured,'-r') %Ux,Tmeasured,'.b',        'Theoretical Voltage',
legend('Tmeasured','Theoretical Temperature','Location','northeast')

figure()               % PLOT ERROR VOLTAGE VS TEMPERATURE
hold on, grid minor;
x0 = 100 + 2*(20+width); y0 = 600 - 1*(80+ height);
set(gcf,'units','points','position',[x0,y0,width,height]);
title('Absolute Error Temperature Measurement'); xlabel('Voltage [U] = [V]'); ylabel(['Temperature: [T] = [C',char(176),']']);
plot(Ux,Terror,'-b')
```

```
legend('Absolute Error','Location','northeast')
```

Appendix 11: Linear potentiometer uncertainty

This appendix aims to provide insight into uncertainty that is caused by the specific sensor and measurement setup as implemented in the proposed Therminus device.

Lineair Potentiometer Uncertainty

The linear potentiometer (LP sensor) setup and its resistor values (in order to maximize the voltage range) is described in the Electrical Circuit Appendix. The values that are given in this specific Appendix are calculated on the basis of a 12bit logic board. Note that the Arduino used in the current Therminus device might not have 12bit read capability on analog read ports. Hence an Arduino with 12bit read capabilities is recommended.

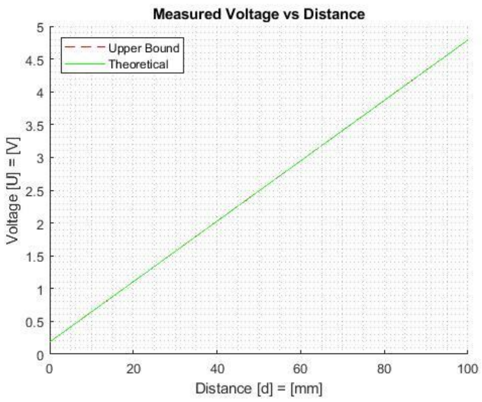


Figure (1): Calibrated lineair potentiometer upper bound voltage at real distance

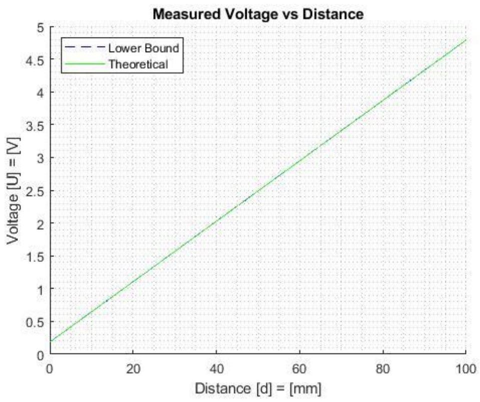


Figure (2): Calibrated lineair potentiometer lower bound voltage at real distance

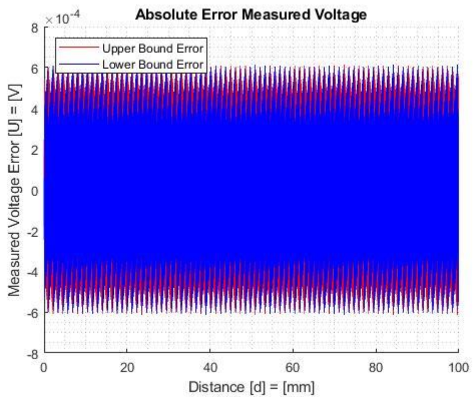


Figure (3): The absolute error resulting from the rounding error caused by discrete Arduino values at specific thickness

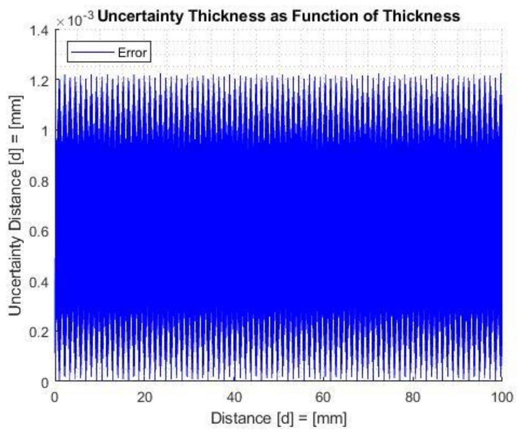


Figure (4): Thickness uncertainty as caused by Arduino rounding error for thickness range

Important values from the script are extracted and displayed to the user. The MATLAB output is given at the end of the appendix alongside the actual MATLAB script.

Parameters as Used in Error Propagation Appendix

To determine the maximum thickness error $\sigma_{d_{textile}}$. The linear potentiometer, by default, is not calibrated which causes it to have a certain precision. To attain the highest possible accuracy, the LP sensor is calibrated. The maximum absolute error is caused by the combination of the discrete voltage measurement (by Arduino) the voltage read to resistance mapping, and the resistance to thickness mapping. Please note that errors caused by floating point approximation are not considered.

Since the error is the result of a number of different factors, the maximum error will be of only concern. The absolute value for maximum error is $\sigma_{d_{textile}} = 1.221e-03$ mm.

MATLAB Script Outputs

```
Measured Voltage [Umax, Umin]: [4.7889, 0.18443]
Theoretical arduino step range [max, min]: [3922, 151]
Useful occupancy of arduino step range: 92.09%

Display maximum uncertainties @ thicknesses [mm]:
Uncertainty [mm] @ Thickness [mm]:       0.001221      23.56
```

```
% Lineair Potentiometer Sensor Error Estimation
% TU Delft, 05/31/2019
% Author: Sebastiaan Thomas Njio

clc
clear variables
close all

Resolution = 2^12-1;

d = 0:2*10^(-5):100*10^(-3); %range of thicknesses

%Declare Constants
Rmax = 10.66*10^(3); Rmin = 18;
Rchosen = 470; %Ohm
Umax = 5*Rmax/(Rchosen +Rmax); Umin = 5*Rmin/(Rchosen +Rmin);

Utheoretical = linspace(Umin, Umax, length(d));
Umeasured = zeros(2,length(Utheoretical));

Steps = (Resolution/5*Utheoretical);
smax = round(Steps(end)); smin = round(Steps(1,1));

%Same mutation that will be performed by arduino
Umeasured(1,:) = 5/Resolution*round(Steps);
Umeasured(2,:) = 5/Resolution*round(Steps);

% Voltage to Thickness Calculation made by Arduino
A1 = d(end) / (smax - smin); B1 = -A1*smin;
% Thickness of the measured material
dmeasurement([1, 2],:) = 1000*[A1*round(Steps) + B1; A1*round(Steps) + B1];



%Errors
C=1:length(d); Error = zeros(2,length(d));
Error(1,:) = (Umeasured(1,:) - Utheoretical(1,:));
Error(2,:) = (Utheoretical(1,:) - Umeasured(2,:));
Error(3,:) = abs(Error(1,:)) + abs(Error(2,:)); %Uncertainty,
% Range of values measured when in reality, d(1,:) should be measured
dikte = 28*10^(-3);
Index2_1 = max(find(d(1,:)<=dikte -0.3*10^(-3))); Index2_2 = max(find(d(1,:)<=dikte-0.3*10^(-3)));
[StepUncertainty(1,1), Index1] = max(Error(3,:));
[StepUncertainty(2,1), Index2] = max(Error(3,1:Index2_2));
%Find max error for 0.45 < d < 28 mm
```

```
StepUncertainty(1,2) = d(1,Index1)*1000;
StepUncertainty(2,2) = d(1,Index2)*1000;
Text = zeros(2,1); Text = char(Text);
Text = ['Uncertainty [mm]:         '; '@ Thickness [mm]:         '];

clear Index1 Index2 Index2_1 Index2_2
dplot = d*1000;
%% Output values
disp(['Measured Voltage [Umax, Umin]: [',num2str(Umax),', ', num2str(Umin),']'])
disp(['Theoretical arduino step range [max, min]: [',num2str(smax),', ', num2str(smin),']'])
disp(['Useful occupancy of arduino step range: ',num2str((smax-smin)/Resolution*100,4),'%']); fprintf('\n')
disp(['*Note that the provided range depends on the orientation of the device.'])
disp(['Display maximum uncertainties @ thicknesses [mm]: '])
disp([Text, num2str(StepUncertainty,4)])

%% Figures

%                                   ///        FIRST FIGURES        \\\
figure()
hold on, grid minor;
x0 = 100; y0 = 600; height = 300; width = 400;
set(gcf,'units','points','position',[x0,y0,width,height]);
title('Measured Voltage vs Distance'); xlabel('Distance [d] = [mm]'); ylabel('Voltage [U] = [V]');
plot(dplot, Umeasured(1,:), '--r', dplot, Utheoretical,'-g')
legend('Upper Bound','Theoretical','Location','northwest')

figure()
hold on, grid minor;
x0 = 100; y0 = 600 - (80+height);
set(gcf,'units','points','position',[x0,y0,width,height]);
title('Error Measured Voltage'); xlabel('Distance [d] = [mm]'); ylabel('Measured Voltage: [U] = [V]');
plot(dplot, Error(1,:), '-r', dplot, (Error(2,:)), '-b')
legend('Upper Bound Error','Lower Bound Error','Location','northwest')



%                                   ///       SECOND FIGURES       \\\

figure()
hold on, grid minor;
x0 = 100 + (20+width); y0 = 600;
set(gcf,'units','points','position',[x0,y0,width,height]);
title('Temperature Vs Voltage'); xlabel('Distance [d] = [mm]'); ylabel([' Voltage [U] = [V]']);
plot(dplot, Umeasured(2,:), '--b', dplot, Utheoretical,'-g')
legend('Lower Bound','Theoretical','Location','northwest')

figure()
hold on, grid minor;
x0 = 100 + 1*(20+width); y0 = 600 - 1*(80+ height);
set(gcf,'units','points','position',[x0,y0,width,height]);
title('Uncertainty Thickness as Function of Thickness');xlabel('Distance [d] = [mm]'); ylabel('Uncertainty Distance [d] =
[mm]');
plot(dplot, Error(3,:) , '-b')
legend('Error','Location','northwest')

%                                   ///       TERTIARY FIGURES       \\\
figure()
hold on, grid minor;
x0 = 100 + 2*(20+width); y0 = 600;
set(gcf,'units','points','position',[x0,y0,width,height]);
title('Uncertainty As Function Of Thickness'); xlabel('Distance [d] = [mm]'); ylabel('Uncertainty Distance [d] = [mm]');
plot(dplot, Umeasured(1,:), '--r', dplot, Utheoretical,'-g', dplot, Umeasured(2,:), '--b')
legend('Upper Bound','Theoretical','Lower Bound','Location','southeast')
```

Appendix 12: Heat loss insulation uncertainty

This Appendix describes the proposed linear interpolation to estimate the real heat loss to the environment through the insulation as function of measured environment temperature.

Heat Loss Interpolation

Appendix 5: 'Insulation' proposes a model for estimating important parameters inherent to the insulation design. The energy balance is too complicated in order to be solved analytically. As mentioned in Appendix Insulation, the energy balance is solved by means of a numerical solver. To reduce the required floating point operations necessary for estimating the heat flow through the insulation, the linear interpolation model is proposed. From Appendix 9: 'Thermal conductivity measurement uncertainty' a maximum error is attained in order not to exceed the 8% measurement accuracy requirement as stated in Appendix 2: 'Phoebe experiment'.

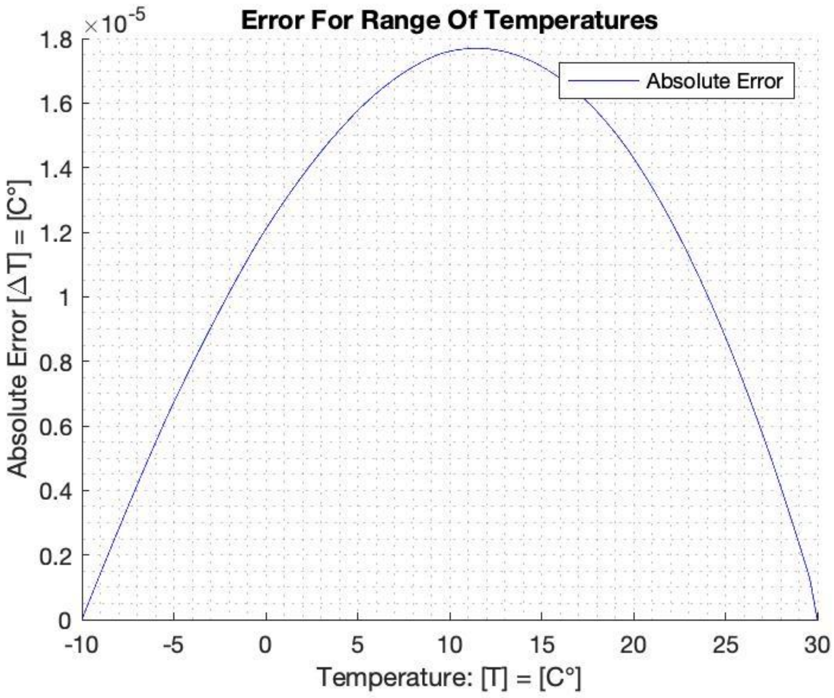


Figure 1: Graph of the absolute temperature error versus the temperature

Important values from the script are extracted and displayed to the user. The MATLAB output, which outputs the linear interpolation values, is given at the end of the appendix alongside the actual MATLAB script.

Parameters as Used in Error Propagation Appendix

The maximum heat flow error will in theory σ_Q be caused by the linear interpolation model. The values for linear interpolation are calculated from the real values from the MATLAB script as provided in Appendix 5: 'Insulation'. The final interpolation values are calculated by the MATLAB script that is provided at the end of this specific Appendix. Note that the provided script loads certain variables from the insulation heat loss script. Thus requiring a line of code to save said variables to a separate file. The resulting value for $\sigma_Q = 1.7694e - 05$ which is represented by the maximum absolute error (error = |proposed value - real value|). Note that this error occurs at an environment temperature of 11.48 C°.

The aforementioned line of code that is required to be run:

```
save('Qis_0.057_28.mat', 'reg','Qmeeting','Qisgok','Tinf_vec')
```

Outputs by MATLAB script

```
Max Heat Flow Insulation [Qismax, @T]: [0.0783686          30]
Min Heat Flow Insulation [Qismin, @T]: [0.077489         -10]

Proposed Lineair Fit for Isolation Heat Flow (Q = A*T + B): (Q = 2.199e-05*T + 0.077709)
Maximum Error Caused by Approximation [Error, @T]: [1.7694e-05, 11.48]
Heat flow @max Error [Qis_linfit, @T]: [0.077961, 11.48]
```

MATLAB Code

```
% Insulation Heat Flow Linear Interpolation for Arduino Application
% TU Delft, 05/31/2019
% Author: Sebastiaan Thomas Njio

clc
close all
clear variables

%% Read data
load('Qis_0.057_28.mat', 'reg','Qmeeting','Qisgok','Tinf_vec')
Qis = reg(4,:);
Qis_simple = Qisgok;
Tinf_vec=Tinf_vec-273.15;

clear reg Qisgok %Clear unnescessary variables in order to prevent workspace clutter

%% Calculations
[Qismax, indexmax]=max(Qis);
[Qismin, indexmin]=min(Qis);
Tinf_vec(1,indexmax);

A = (Qismax - Qismin)/(Tinf_vec(1,indexmax)-Tinf_vec(1,indexmin));
B = Qismax - A*Tinf_vec(1,indexmax);

Qis_linfit = A.*Tinf_vec + B;
Qis_error = Qis - Qis_linfit;
[Qis_errormax, indexerrormax] = max(Qis_error);

% Display values to user
disp(['Max Heat Flow Insulation [Qismax, @T]: [',num2str([Qismax, Tinf_vec(1,indexmax)]),']']);
disp(['Min Heat Flow Insulation [Qismin, @T]: [',num2str([Qismin, Tinf_vec(1,indexmin)]),']']); fprintf('\n');
disp(['Proposed Lineair Fit for Isolation Heat Flow (Q = A*T + B): (Q = ',num2str(A),'*T + ',num2str(B),')'])
disp(['Maximum Error Caused by Approximation [Error, @T]: [',num2str(Qis_errormax),',
',num2str(Tinf_vec(indexerrormax)),']'])
disp(['Heat flow @max Error [Qis_linfit, @T]: [',num2str(Qis_linfit(indexerrormax)),',
',num2str(Tinf_vec(indexerrormax)),']'])

%                         ///       FIRST FIGURES       \\\
figure()            % PLOT HIGH FIDELITY VOLTAGE VS TEMPERATURE
hold on, grid minor;
x0 = 100; y0 = 600; height = 300; width = 400;
set(gcf,'units','points','position',[x0,y0,width,height]);
title('title'); xlabel('xlabel'); ylabel('ylabel')
plot(Tinf_vec, Qis,'-b',Tinf_vec,Qis_simple*ones(length(Tinf_vec)),'-g',Tinf_vec,Qis_linfit,'r')
legend('Real Qis','Simple Qis','Interpolation Qis','Location','northeast')
```

```
figure()                % Error
hold on, grid minor;
x0 = 100; y0 = 600 - (80+height);
set(gcf,'units','points','position',[x0,y0,width,height]);
title('Error For Range Of Temperatures');xlabel(['Temperature: [T] = [C',char(176),']']); ylabel(['Absolute Error [\DeltaT] =
[C',char(176),']']);
plot(Tinf_vec, Qis_error,'-b')
legend('Absolute Error','Location','northeast')
```

Appendix 13: General project planning

A general outline of the project planning was made at the beginning of the project, with a generic description of the different milestones within the project. The planning consists of 9 tasks. A more detailed description of the planning is displayed in table 1.

1. Final concept choice (2 days): e.g. *evaluation criteria*
2. Design (e.g. parts) (1 week)
3. Design Freeze
4. Research / test samples in the lab (phoebe) (3 days)
5. Detailed design (frame etc.) (1 week)
6. Make prototype (3 days)
7. Test / validate / calibrate (1 week)
8. Report
9. Presentation

Table 1: A detailed overview of the project planning

Week	Moment	Events (day)	Activities
1	March 25-29		Final Concept Selection
2	April 1-5		Study PHOEBE+contact lab
3	April 8-12	Exams	
4	April 15-19	Exams	Selection of parts (Wednesday 17)
5	April 22- 26		Design, Design freeze, part selection
6	April/May 29-3		Lab tests
7	May 6-10		First prototype and Tests
8	May 13-17	Interim Report II (16)	Tests and Validation
9	May 20-24		Final Report
10	May 27-31		Allocated for delays
11	June 3-7	Final Paper Deadline (7)	Allocated for additional delays
12	June 11-14	Oral Exam (12)	
13	June 17-21	Final presentation (20)	

Appendix 14: Clothing thickness dataset

A dataset is created of thicknesses of clothing pieces, by measuring the thickness of 25 winter coats, 25 summer coats, 25 sweaters and 25 T-shirts with a caliper. Based on this dataset, a good estimation is made on the average thickness of clothing pieces as well as the variance and minimum and maximum thickness. This information is used for calculations and experiments. The gathered information is shown in table 1.

Table 1: Dataset of thicknesses of clothing pieces.

Piece of clothing	Winter coat	Summer coat	Sweater	T-shirt
Set 1 (Average)	3.34	1.096	1.77	0.76
1	3.45	0.69	2.35	0.6
2	3.05	2.5	1.3	0.95
3	2	0.78	1.2	0.85
4	2.8	0.765	1.95	0.8
5	5.4	0.745	2.05	0.6
Set 2 (Average)	10.102	1.08	2.02	1.042
1	8.16	1.3	0.75	1.03
2	12.7	1.1	1.25	1.27
3	8.55	0.8	3.8	0.91
4	6.8	1.2	3.1	1.11
5	14.3	1	1.2	0.89
Set 3 (Average)	8.448	2.8512	4.898	0.93
1	11.42	3.5	4.29	1.12
2	5.89	1.7	6.33	1.59
3	7.48	4.7	7.7	0.47
4	10.41	1.98	2.59	0.25
5	7.04	2.376	3.58	1.22
Set 4 (Average)	7.2	0.6	1.76	0.58
1	6	0.4	1.5	0.4
2	10	0.4	1.7	0.7
3	7	0.7	2.5	0.4
4	7	1.1	1.7	0.5
5	6	0.4	1.4	0.9
Set 5 (Average)	14.308	0.61	2.64	0.67
1	5.24	0.5	3	0.5
2	12	0.6	4.3	0.6
3	14.3	0.3	2.7	1.3
4	4.6	0.2	2	0.6

5	35.4	1.45	1.2	0.35
Average of all sets	8.6796	1.24744	2.6176	0.7964
Max	35.4	4.7	7.7	1.59
Min	2	0.2	0.75	0.25
Variance	42.91	1.14	2.78	0.12