Appendix

1 Project brief

DESIGN FOR our future

TU Delft

Personal Project Brief – IDE Master Graduation Project

Name student Manon van der Stap **Student number** 5,255,996

PROJECT TITLE, INTRODUCTION, PROBLEM DEFINITION and ASSIGNMENT
Complete all fields, keep information clear, specific and concise

Project title knie-wearable for rheumatic patients

Please state the title of your graduation project (above). Keep the title compact and simple. Do not use abbreviations. The remainder of this document allows you to define and clarify your graduation project.

Introduction

Describe the context of your project here; What is the domain in which your project takes place? Who are the main stakeholders and what interests are at stake? Describe the opportunities (and limitations) in this domain to better serve the stakeholder interests. (max 250 words)

This project takes place in the healthcare and smart textiles domain, specifically focusing on the detection and monitoring of osteoarthritis (OA) using wearable technologies. OA is a chronic degenerative joint disease characterized by cartilage loss, pain, and stiffness, conditions that require long-term management and personalized care.

The primary stakeholders are rheumatic patients, clinicians, and researchers. For patients, the interest lies in reducing pain and improving mobility through interventions. Clinicians seek reliable, real-time data to support diagnosis and tailor treatment, while researchers aim to develop non-invasive tools that enhance understanding of disease progression.

This project explores the use of smart wearables to measure two physiological OA markers: skin temperature and joint acoustics. These markers can indicate local inflammation and cartilage degeneration. The opportunity lies in developing a comfortable, monitoring solution that captures these subtle signals in daily life. Such tools could empower patients, reduce clinical visits, and provide detailed datasets for analysis.

However, limitations include variability in temperature due to external factors, signal noise in dynamic environments, and the need for personalized baseline measurements. The integration of acoustic and thermal sensors in wearables has the potential to fill a gap, by clearly measuring, in a non-invensive way, the onset and progression of symptoms in the day to day life of a patient.

One existing brace, developed by SenseModi, is relatively bulky and visible under clothing (figure 1 & 2). A non-invasive and more discreet alternative would be a logical next step in its development.

→ *space available for images / figures on next page*

introduction (continued): space for images

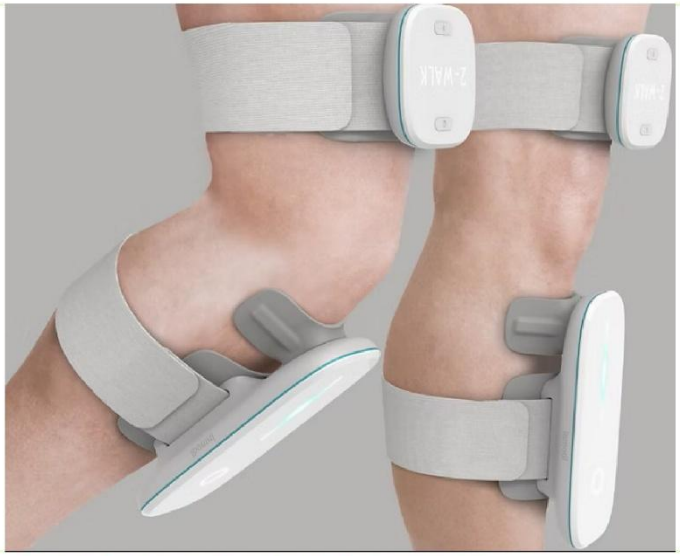


image / figure 1 Sensemodi, Smart Brace: Joint monitoring for healthcare providers

image / figure 2 Sensemodi smart brace with pants over it

Personal Project Brief – IDE Master Graduation Project

Problem Definition

What problem do you want to solve in the context described in the introduction, and within the available time frame of 100 working days? (= Master Graduation Project of 30 EC). What opportunities do you see to create added value for the described stakeholders? Substantiate your choice.
(max 200 words)

Osteoarthritis (OA) is a progressive joint disease. Current diagnostic methods, such as radiographic imaging, are limited in detecting early symptoms and offer only momentary snapshots of joint condition. Patients and clinicians lack access to real-time, continuous data that could support better disease management and early intervention. This project focuses on creating a wearable that can detect signs of OA during daily activities, making symptom tracking more natural and consistent, skin temperature and joint acoustics. By focusing on these signals, the project seeks to provide a non-invasive, real-time solution for monitoring joint health.

In the scope of 100 working days, this project will explore the technical feasibility of using conductive yarns or NTC Sensors and contact microphones for sensing, supported by signal processing techniques. It will also assess usability in a clinical or daily-life context.

The added value lies in empowering patients with better symptom awareness, reducing healthcare costs through early detection, and providing clinicians with objective, longitudinal data to adapt treatment strategies. This project bridges a gap between medical needs and wearable health technology innovation.

Assignment

This is the most important part of the project brief because it will give a clear direction of what you are heading for. Formulate an assignment to yourself regarding what you expect to deliver as result at the end of your project. (1 sentence) As you graduate as an industrial design engineer, your assignment will start with a verb (Design/Investigate/Validate/Create), and you may use the green text format:

Design and Create a non-invasive working knee wearable prototype to measure acoustic emissions and temperature changes for rheumatic patients and doctors to get a better understanding of complaints.

Then explain your project approach to carrying out your graduation project and what research and design methods you plan to use to generate your design solution (max 150 words)

The graduation project will follow an iterative, user-centered design process. It will begin with qualitative research, including interviews with osteoarthritis patients and medical professionals, to identify needs, experiences, and opportunities. Also research will be done into the development or choise of the sensors used in the wearable.

Based on the findings, initial low-fidelity prototypes of the wearable system will be developed to explore different sensing configurations. These will be refined into functional prototypes using conductive yarns (or other sensors) and microphones, which will undergo testing for usability, comfort, and signal quality.

Validation will include experiments and, if feasible, user trials to evaluate performance in realistic scenarios. Signal processing techniques will be used to analyze the data and assess reliability of physiological markers. Throughout, continuous feedback loops with stakeholders will ensure that the design remains relevant, feasible, and valuable in clinical and daily-life contexts.

Project planning and key moments

*To make visible how you plan to spend your time, you must make a planning for the full project. You are advised to use a Gantt chart format to show the different phases of your project, deliverables you have in mind, meetings and in-between deadlines. Keep in mind that all activities should fit within the given run time of 100 working days. Your planning should include a **kick-off meeting**, **mid-term evaluation meeting**, **green light meeting** and **graduation ceremony**. Please indicate periods of part-time activities and/or periods of not spending time on your graduation project, if any (for instance because of holidays or parallel course activities).*

Make sure to attach the full plan to this project brief.
The four key moment dates must be filled in below

Kick off meeting	18 Sep 2025
Mid-term evaluation	19 Nov 2025
Green light meeting	18 Feb 2026
Graduation ceremony	27 mrt 2026

In exceptional cases (part of) the Graduation Project may need to be scheduled part-time. Indicate here if such applies to your project

Part of project scheduled part-time	✔
For how many project weeks	25
Number of project days per week	4,0

Comments:

Motivation and personal ambitions

Explain why you wish to start this project, what competencies you want to prove or develop (e.g. competencies acquired in your MSc programme, electives, extra-curricular activities or other).

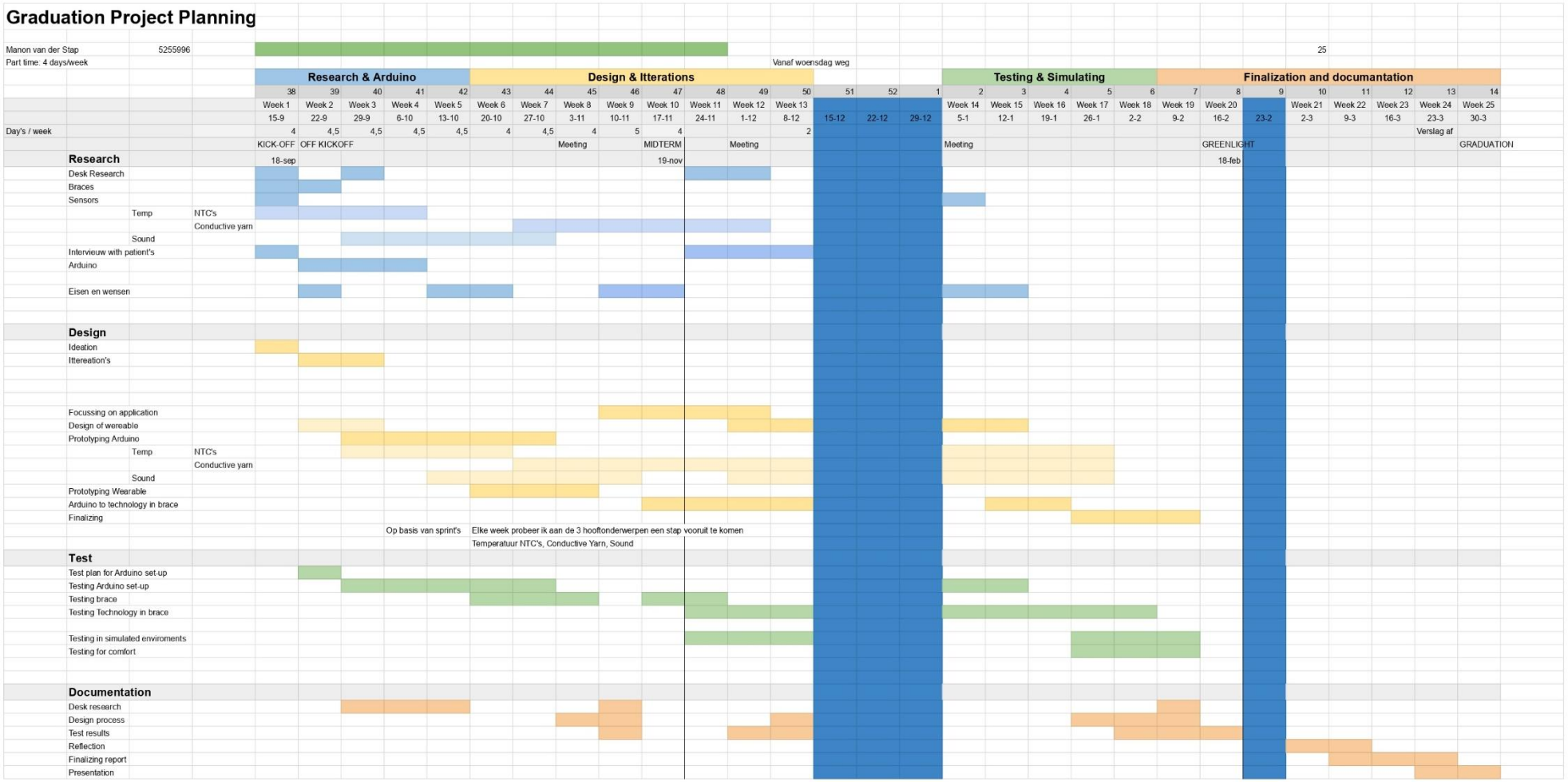
Optionally, describe whether you have some personal learning ambitions which you explicitly want to address in this project, on top of the learning objectives of the Graduation Project itself. You might think of e.g. acquiring in depth knowledge on a specific subject, broadening your competencies or experimenting with a specific tool or methodology. Personal learning ambitions are limited to a maximum number of five.
(200 words max)

I am motivated to start this project because it offers a clear direction from the beginning, allowing me to work efficiently toward a meaningful result. I am particularly interested in gaining deeper knowledge of sensor technologies and understanding how these are applied in real-life medical contexts. The opportunity to explore both the technical and human aspects of this challenge, combining engineering with patient need, matches my ambitions.

Since a young age, I have been fascinated by textiles and spent hours behind the sewing machine, exploring materials and how they interact with the body. With this project, I see an opportunity to combine that early creative interest with my current technical design skills. Combining soft materials and sensor integration feels like an intresting direction.

My personal goal is to develop a functional prototype that is not only technical but also tested and validated by real users. Being able to test it in a realistic medical setting and possibly improve lives feels like a valuable goal. This project aligns perfectly with my interest in healthcare innovation and allows me to complete my studies with impact and new skills, while deepening my understanding of both design and medical technology.

Table 1: Overvieuw of project planning



2 Panel

2.1 Setup

Introduction to the project

1. Introduction

- Brief explanation of the project and the purpose of the wearable:
- Explanation that it is based on assumptions and that it can provide insight into the progression of crepitus and knee temperature. (Emphasise that the design is based on an assumption)
- Explanation of how the patient would ultimately benefit from this wearable:
- If the hypothesis were true, this could mean that increased crepitus activity and temperature rise could prompt early medication, thereby suppressing a flare-up and causing less damage. This would mean less pain.
- Explain that this is an exploratory session: we are interested in their initial reactions and ideas. Critical input and thinking aloud are important.
- Question: 'What are your initial thoughts on this? Feel free to share whatever comes to mind.'

2. Concept questions

- What do you think of the idea of a wearable device that continuously measures crepitus and temperature?
- What would you like such a wearable device to look like?
    - What should it definitely include?
    - What would you rather not have?

3. Integration into daily life

- Explain that for accurate measurement, it is important to measure throughout the day:
- What do you think about this? At what times would you wear it, and when would you not wear it?
- Note down the times of the day when you would wear the wearable and when you would not. Why or why not?

4. Show Parts of wearable

- Prototypes (Concept 4, Concept 5, Sleeve Concept) (Appendix figure 2)
- Hardware (3D printed) and hardware placement (Appendix figure 4)
- Fabric options (Appendix figure 3)
- Colour options (Appendix figure 1)

Ask what they think of it, and which one they prefer, and have an open discussion

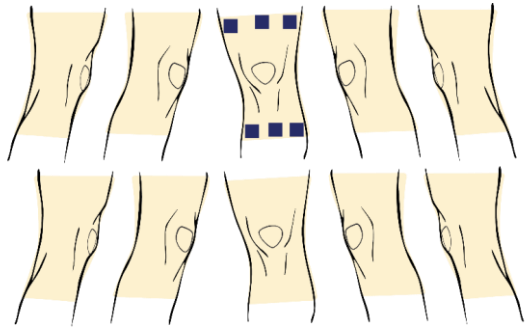
5. Ask for other notes.

2.2 Figures shown

Appendix figure 2: Concepts shown, Knee sleeve. Retrieved from https://incrediwear.eu/products/incrediwear-knee-sleeve other: Self-taken

Appendix figure 1: Colour options

Appendix figure 3: Fabrics evaluated



Appendix figure 4: Placement hardware

2.3 Responses panel

Table 2: Responses of panel categorised per part

Person	Placement hardware	Concept preferance	Color	Fabric	Notes
1	Best position is where as little movement is as possible	Concept 2 is the best. Does not like the band over the patella	Does not matter, price is more important	Fabric 2, Soft and strong	
2	Lower left or right	Concept 1 and 2 look tight around open parts, Concept 3 looks the most comfortable	skin color, black or gray, price is also important, maybe younger people like color more?	Fabric 1, is the most flexible (preference) Fabric 2, do not like Fabric 3, Is to thin you might feel the sensors through the fabric? Fabric 4, on the second place	
3	Upper, middle	Concept 2, breathable fabric is important, the wearable does not need to be to thigh, but have to keep in place.	Black is fine	Fabric 1 is he most stretchy	
4	Upper, middle, outside	You can linger in the holes of the first two concepts, concept 3 looks like it will obstruct movement.	-	Fabric 4, is an irritationg fabric and keeps smell, washing needs to be done more frequently, Fabric 3, is lightly irriting, Fabric 1, is less irritating, Fabric 2, is the best fabric rolls up a bit.	
5	Upper, outside	Concept 2, is a breathable concept, however there are certain pressure points. Concept 3 is to much fabric, however pressure is spread.	Skin tone, so it is not as noticeable, otherwise one color is good, maybe for kids multiple colors available.	Fabric 3 feels the nicest, it is light, and stretchy.	Extra: maybe for signaling pain or symptoms you can make a "screenshot" so the device can show when pain is observed by the user.
6	Lower, outside	Concept 2, is a nice open design, Concept 3 looks to warm, Concept 1 looks like it is to tight on the back of the knee.	Color does not matter for me (but I am a man), However making it clear what is the upper side would be nice.	Fabric 3, Make the fabric flexible, so people can choose for themselves.	
7	Upper, back	Concept 1, the band is to tight on the back of the knee, Concept 3, is not so breathable. Concept 2 the bands look a bit thin.	Color does not matter for me.	Fabric 4 is the nicest, 1 as second, 2 as 3th, and 3 as last.	
8	upper, outside	Concept 1, looks most comfortable, your toes will probably go through the holes when putting it on. Concept 3, looks to warm and will probably stink.	Color will depend per person, per situation, maybe some standard colors as options?	Fabric 3, is stretchy and breathable.	Maybe working with an SD card is better, because WiFi or Bluetooth need constant connection.
Other notes	Placement where there is as little momvement as possible.	Concept 2 your patella is free which seems more comfortable.	Fun color is nice, but scintone is also good. Production price is importand, as long as it looks presentable it is alright.	Fabric 3, is really strechy, and thin will I feel the sensors?	

Table 3: Responss of panel other questions

What do you think of wearing it for 16 hours per day?	It is the same as a bra for me. Seems a bit long. If it is temporary it is fine for me. Depends on the comfort level. If it makes sure I have less pain I am up for it.
When would you wear it?	If I know the pain will be less, and the wearable wil not irritate me with warmth. I also want to gain something from it. Think about winter and summer differences in temperature.
Other	Different colors for upside and downside of the wearable, User interface will probably be different for younger people than older people. When you have hairy skin the grip elastic does not work as good. Breathability is important however you need to look out for the fabric cutting the skin, and for rolling of the fabric. Washing is also important, when I will wear it for most of the days

3 Comfort test of concept 8

3.1 Test opzet

Doel van de test: Het doel van deze draagtest was het valideren van het ergonomisch ontwerp, de stabiliteit en het thermisch comfort van concept 8 gedurende een langere periode. Er werd specifiek getoetst of de wearable voldoet aan de vastgestelde gebruikersvereisten (User requirements) in een realistische, alledaagse context. Er werdt vooral gekeken naar mogelijke verbeteringen.

Deelnemer en concept: 1 gezonde volwassene. Concept 8

Procedure: De test bestond uit een continue draagperiode van 4 uur. De procedure verliep als volgt:

1. **Aantrekken:** De gebruiker werd gevraagd de wearable aan te trekken, waarbij de intuïtiviteit en de benodigde tijd (< 60 seconden) werden geobserveerd.
2. **Positiemarkering:** De exacte positie van de wearable op het been werd gemarkeerd om latere verschuivingen te kunnen meten.
3. **Draagfase:** De deelnemer droeg de wearable gedurende 4 uur onder een reguliere lange broek tijdens alledaagse activiteiten (lopen, zitten etc.).
4. **Eindmeting en interview:** Na 4 uur werd de positieverschuiving opgemeten. Aansluitend vond een semi-gestructureerd interview plaats, gebaseerd op de ontwerpeisen rondom comfort, stabiliteit en gebruiksgemak.

3.2 Test resultaten

De resultaten van het afsluitende interview zijn gecategoriseerd op basis van ontwerpcriteria.

Draagcomfort en pasvorm

- **Draagervaring:** De wearable werd als zeer comfortabel ervaren. Na verloop van tijd werd het dragen nauwelijks meer opgemerkt (de gebruiker gaf het een score van 2 op 10 qua 'aanwezigheid').
- **Drukpunten:** Er werden geen noemenswaardige of pijnlijke drukpunten ervaren. Echter voelde het elastiek aan de bovenrand na verloop van tijd wat strak aan, wat na het uittrekken resulteerde in lichte rode afdrukken op de huid.
- **Bewegingsvrijheid:** Bij het volledig buigen van de knie was de stof voelbaar onder de knieschijf, maar dit werd niet als belemmerend of storend ervaren.
- **Onder kleding:** De gladde stof zorgde ervoor dat een lange broek er gemakkelijk overheen gleed, zonder wrijving of opkruipen te veroorzaken.

Stabiliteit en thermisch comfort

- **Stabiliteit:** Hoewel de gebruiker tijdens het lopen soms het gevoel had dat de wearable afzakte, bleek dit in de praktijk niet zo te zijn. Na 4 uur dragen was de fysieke verschuiving slechts 0,5 cm.
- **Vormbehoud:** Aan de zijkanten van de knie "flubberde" de stof enigszins. Bij het strekken en buigen frummelde of vouwde de stof op deze plekken wat op.
- **Warmte / ademend vermogen:** De knie werd niet overmatig warm en er was geen sprake van zweet. Het open ontwerp (waarbij niet de hele knie wordt bedekt) droeg hier zeer effectief aan bij.

Gebruiksgemak

- **Proces:** Het aantrekken werd vergeleken met het aantrekken van een sok of panty. De gebruiker merkte op voorzichtig te werk te gaan om het materiaal niet te scheuren bij de uitsparingen. Er is behoefte aan een soepeler aantrekproces.
- **Oriëntatie:** Het was voor de gebruiker niet direct intuïtief wat de boven- of onderkant was. Het gat voor de patella (knieschijf) diende als een duidelijk visueel ankerpunt, maar er is sterke behoefte aan extra visuele indicatoren (zoals een pijl of label).
- **Tijdsduur:** Zodra de juiste handigheid was gevonden, lukte het aantrekken moeiteloos binnen de gestelde eis van 60 seconden.

4. Aanbevelingen van de gebruiker (Verbeterpunten voor design)

- **Afwerking:** Om irritatie te voorkomen en de esthetiek te verbeteren, wordt het gebruik van "no-seam" (naadloze) technieken aangeraden.
- **Aansluiting:** Het netjes omzomen van de randen of verbeteren van het patroon kan helpen om de wearable (vooral aan de zijkanten) strakker en mooier op het been te laten aansluiten, wat het 'flubberen' tegengaat.
- **Anatomische vormgeving:** De uitsparingen (gaten) zouden in de toekomst nog specifieker afgestemd kunnen worden op de spiergroepen die daadwerkelijk aanspannen tijdens de beweging.
- **Visuele cues:** Het toevoegen van een duidelijke 'bovenkant'-indicator is essentieel voor sneller en foutloos gebruik.

4 Addition test

4.1 Test opzet

Doel van de test: Het doel van deze test was het vergelijken van het draagcomfort, het aantrekgemak (donning) en het fysieke comfort van twee prototype-iteraties. Specifiek was de test erop gericht om vast te stellen of het toevoegen van een dunne laag gaasstof (panty-materiaal) over de uitsparingen de gebruikerservaring verbeterde ten opzichte van het standaard open-gaten ontwerp.

Deelnemers

- **Steekproefgrootte:** 5 deelnemers.
- **Proefpersonen:** gezonde jongvolwassenen (medestudenten).

Materialen en prototypes

- **Concept 8:** De wearable met open uitsparingen.
- **Concept 8 with addition:** Dezelfde wearable, waarbij de uitsparingen waren bedekt met een dunne laag pantystof/gaas.

Procedure De testsessies werden individueel uitgevoerd in een gecontroleerde binnenomgeving. Het protocol bestond uit de volgende stappen:

1. **Introductie:** Deelnemers kregen uitleg over het doel van de wearable, maar ontvingen vooraf géén instructies over hoe ze deze moesten aantrekken. Dit werd gedaan om de intuïtie van de gebruiker te observeren en knelpunten te identificeren.
2. **Taak 1 (Aantrekken concept 8):** De deelnemer werd gevraagd de standaard open-gaten wearable aan te trekken. De onderzoeker observeerde of de voet ergens bleef haken en of de oriëntatie van de band verwarrend was.
3. **Taak 2 (Aantrekken concept 8 met toevoeging):** De deelnemer werd gevraagd de vorige wearable uit te trekken en de iteratie met de pantystof aan te trekken.
4. **Kwalitatieve evaluatie:** Terwijl de prototypes gedragen werden (en tijdens beweging), werd er een semi-gestructureerd interview afgenomen gericht op de vergelijking, het comfort.
5. **Interviewleidraad / evaluatiecriteria** De volgende vragen dienden als leidraad voor de kwalitatieve feedbacksessie:

    - **Aantrekgemak:** Maakte de toevoeging van de pantystof het duidelijker of makkelijker om de wearable aan te trekken? Waren er minder momenten waarop je voet bleef haken?
    - **Vergelijkend comfort:** Als je de twee iteraties vergelijkt, wat zijn dan de specifieke voor- en nadelen met betrekking tot het tastgevoel op de huid en de pasvorm?
    - **Thermische perceptie:** Hoe verhoudt het thermisch comfort (warmte en ademend vermogen) van het open-gaten concept zich tot het panty-concept?
    - **Algehele voorkeur:** Welk concept heeft over het algemeen je voorkeur, en waarom?

4.2 Test resultaten

Testresultaten: Gecategoriseerde Gebruikersfeedback (P1 - P5)

Aantrekgemak

- **P1:** Het panty-concept was iets makkelijker aan te trekken omdat de gaten dicht zijn; je voeten blijven er niet achter hangen. Bij het

originele concept met de open gaten viel de voet er in het begin sneller verkeerd in (omdat je "meer opties" hebt).

- **P3:** *"Much easier, because there were less points of error. Less holes."*
- **P4:** Omdat er minder gaten waren, ging het aantrekken automatisch. Twee dichte gaten helpen daar heel erg bij. Echter: als je eenmaal weet hoe hij aan moet, kost het aantrekken weinig moeite en heb je de panty eigenlijk niet meer nodig.

Vergelijkend Comfort

- **P1:** De panty heeft een lekker zachte stof.
- **P3:** *"It feels okay. The panty is nice, but it has more fabric and is warmer."*
- **P4:** Door de extra spanning en het stiksel aan de kant van de panty, sluit de wearable wel beter en strakker aan om het been. Er zit meer stof, wat iets minder is, maar het stoort niet per se.
- **P5:** De afwerking kan beter: mag er over het algemeen wat steviger uitzien, maar het is ook nog maar een prototype. De toegevoegde pantystof is zacht en superdun, maar absoluut niet stevig.

Thermische perceptie

- **P1:** De panty-stof zou in de winter wel lekker warm zijn, maar voor in de zomer is de extra laag waarschijnlijk te warm.
- **P5:** Het origineel zonder panty ademt veel meer. Met de panty is hij merkbaar warmer.

Algehele voorkeur & aanbevelingen van gebruikers

- **P1:** De eerste (zonder panty) is beter qua gevoel. Aanbeveling: liever een gebruiksaanwijzing maken voor concept 1, dan de pantystof toevoegen.
- **P4:** Als de juiste aantrektechniek eenmaal bekend is, is het makkelijk genoeg en is de extra stof overbodig.
- **P5:** Voorkeur gaat uit naar de versie *zonder* panty. Deze zit over het algemeen fijner, ademt meer, en het grote nadeel van de pantystof is dat deze te fragiel is (je trekt er zo een ladder in).

5 Placement of the acoustic sensor

5.1 Python code

```
import wave
import numpy as np
import matplotlib.pyplot as plt
import os
from scipy.signal import butter, filtfilt, resample, hilbert

# ========================================
# CONFIGURATION
# ========================================
FILES = [
    "Left-withoutTrousers.wav",
    "Left-withTrousers.wav",
    "Right-withoutTrousers.wav",
    "Right-withTrousers.wav"
]

LOCATIONS = [
    "Medial - without Trousers",
    "Medial - with Trousers",
    "Lateral - without Trousers",
    "Lateral - with Trousers"
]

TARGET_FS = 48000
LOWCUT = 10000
HIGHCUT = 20000

# ========================================
# SIGNAL PROCESSING FUNCTIONS
# ========================================

def butter_bandpass(lowcut, highcut, fs, order=6):
    nyq = 0.5 * fs
    low = lowcut / nyq
    high = highcut / nyq
    b, a = butter(order, [low, high], btype='band')
    return b, a

def apply_toreyin_filter(data, fs):
    b, a = butter_bandpass(LOWCUT, HIGHCUT, fs, order=6)
    return filtfilt(b, a, data)

def get_hilbert_envelope(signal):
    analytic_signal = hilbert(signal)
    return np.abs(analytic_signal)
```

```
def load_and_preprocess(file_path):
    if not os.path.exists(file_path):
        return None, None

    with wave.open(file_path, "rb") as wf:
        fs = wf.getframerate()
        n_frames = wf.getnframes()
        audio = np.frombuffer(wf.readframes(n_frames),
dtype=np.int16).astype(np.float32)

        # Geen piek-normalisatie, maar absolute schaling (16-bit bereik)
        audio /= 32768.0

        if fs != TARGET_FS:
            audio = resample(audio, int(len(audio) * TARGET_FS / fs))
            fs = TARGET_FS

    return audio, fs

# ========================================
# PLOTTING FUNCTIONS
# ========================================

def plot_unfiltered_comparison():
    # --- GLOBALE SCHAAL BEPALEN ---
    plot_data = []
    global_max = 0
    window_size_samples = int(TARGET_FS * 0.05) # 50ms window

    for file_name in FILES:
        audio, fs = load_and_preprocess(file_name)
        if audio is not None:
            rms = [np.sqrt(np.mean(audio[j:j+window_size_samples]**2))
                   for j in range(0, len(audio), window_size_samples)]
            t = np.linspace(0, len(audio)/fs, len(rms))

            current_max = np.max(rms)
            if current_max > global_max:
                global_max = current_max

            plot_data.append((t, rms))
        else:
            plot_data.append(None)

    y_limit = global_max * 1.1 if global_max > 0 else 1.0

    # --- AANPASSING: sharey=False zodat ELKE grafiek een as heeft ---
    fig, axes = plt.subplots(2, 2, figsize=(14, 10), sharex=False, sharey=False)
    axes = axes.flatten()

    for i, data in enumerate(plot_data):
        if data is not None:
            t, rms = data
            axes[i].plot(t, rms, color='teal', linewidth=1)
```

```
            axes[i].set_title(LOCATIONS[i])
            axes[i].grid(True, alpha=0.3)

            # Wel de limiet forceren zodat ze vergelijkbaar blijven
            axes[i].set_ylim(0, y_limit)

            # --- AANPASSING: Labels BIJ ELKE GRAFIEK ---
            axes[i].set_xlabel("Time (s)")
            axes[i].set_ylabel("Amplitude (a.u.)")

        else:
            axes[i].text(0.5, 0.5, "File Not Found", ha='center')
            axes[i].set_title(LOCATIONS[i])

    plt.suptitle("Raw Noise Comparison (Unfiltered RMS)", fontsize=16)
    plt.tight_layout(rect=[0.05, 0.05, 1, 0.95])
    plt.show()

def plot_unfiltered_hilbert_comparison():
    # --- GLOBALE SCHAAL BEPALEN ---
    plot_data = []
    global_max = 0

    for file_name in FILES:
        audio, fs = load_and_preprocess(file_name)
        if audio is not None:
            # GEEN filtering, direct Hilbert envelope berekenen op de ruwe data
            envelope = get_hilbert_envelope(audio)

            t = np.arange(len(envelope)) / fs

            current_max = np.max(envelope)
            if current_max > global_max:
                global_max = current_max

            plot_data.append((t, envelope))
        else:
            plot_data.append(None)

    y_limit = global_max * 1.1 if global_max > 0 else 1.0

    # --- AANPASSING: sharey=False ---
    fig, axes = plt.subplots(2, 2, figsize=(14, 10), sharex=False, sharey=False)
    axes = axes.flatten()

    for i, data in enumerate(plot_data):
        if data is not None:
            t, envelope = data

            axes[i].plot(t, envelope, color='#1d3557', linewidth=0.7,
label="Hilbert Envelope")
            axes[i].set_title(f"{LOCATIONS[i]} ")
            axes[i].grid(True, alpha=0.2)
```

```
            # Wel de limiet forceren
            axes[i].set_ylim(0, y_limit)

            # --- AANPASSING: Labels BIJ ELKE GRAFIEK ---
            axes[i].set_xlabel("Time (s)")
            axes[i].set_ylabel("Amplitude (a.u.)")

        else:
            axes[i].text(0.5, 0.5, "File Not Found", ha='center')
            axes[i].set_title(f"{LOCATIONS[i]} ")

    plt.suptitle("Placement Assessment: Unfiltered Broadband Hilbert Envelope",
fontsize=16)
    plt.tight_layout(rect=[0.05, 0.05, 1, 0.95])
    plt.show()

def plot_toreyin_filtered_comparison():
    # --- GLOBALE SCHAAL BEPALEN ---
    plot_data = []
    global_max = 0

    for file_name in FILES:
        audio, fs = load_and_preprocess(file_name)
        if audio is not None:
            filtered = apply_toreyin_filter(audio, fs)
            envelope = get_hilbert_envelope(filtered)

            t = np.arange(len(envelope)) / fs

            current_max = np.max(envelope)
            if current_max > global_max:
                global_max = current_max

            plot_data.append((t, envelope))
        else:
            plot_data.append(None)

    y_limit = global_max * 1.1 if global_max > 0 else 1.0

    # --- AANPASSING: sharey=False ---
    fig, axes = plt.subplots(2, 2, figsize=(14, 10), sharex=False, sharey=False)
    axes = axes.flatten()

    for i, data in enumerate(plot_data):
        if data is not None:
            t, envelope = data

            axes[i].plot(t, envelope, color='#1d3557', linewidth=0.7,
label="Hilbert Envelope")
            axes[i].set_title(f"{LOCATIONS[i]} (10-20 kHz)")
            axes[i].grid(True, alpha=0.2)

            # Wel de limiet forceren
            axes[i].set_ylim(0, y_limit)
```

```
            # --- AANPASSING: Labels BIJ ELKE GRAFIEK ---
            axes[i].set_xlabel("Time (s)")
            axes[i].set_ylabel("Amplitude (a.u.)")

        else:
            axes[i].text(0.5, 0.5, "File Not Found", ha='center')
            axes[i].set_title(f"{LOCATIONS[i]} (10-20 kHz)")

    plt.suptitle("Placement Assessment: Filtered Toreyin and Teague Method",
fontsize=16)
    plt.tight_layout(rect=[0.05, 0.05, 1, 0.95])
    plt.show()

if __name__ == "__main__":
    plot_unfiltered_comparison()
    plot_unfiltered_hilbert_comparison()
    plot_toreyin_filtered_comparison()
```

6 NTC accuracy

6.1 Table's

Table 4: Measurements at room temperature

Round 1, measurement room temperature	Degrees in (°C)	Peak to peak in (°C)	Standard deviation σ (°C)	1 degree range ± 0.5 °C
1	18.3	7.8	0.2504	95.42%
2	18.5	7.8	0.1829	99.37%
3	18.5	7.4	0.2671	93.88%
4	18.8	7.6	0.2533	95.16%
5	18.6	7.7	0.3096	89.37%
6	18.4	7.3	0.2105	98.25%
7	19	7.6	0.2815	92.43%
8	18.6	7.8	0.2305	96.99%
Average	18.6	7.6	0.24523	95.11%

Table 5: Measurements on a hotplate (controlled environment)

Round 2, measurement hotplate 25 °C, controlled	Degrees in (°C)	Peak to peak in (°C)	Standard deviation σ (°C)	1 degree range ± 0.5 °C
1	25.9	3.5	0.1254	99.99%
2	26.1	2.9	0.1116	100%
3	26.1	3.2	0.1106	100%
4	26	3	0.1101	100%
5	25.9	3.2	0.1183	100%
6	25.9	2.9	0.112	100%
7	25.9	3.2	0.1111	100%
8	25.9	2.7	0.1126	100%
Average	25.96	3.1	0.11396	99.99%

6.2 Arduino code

```
#include <math.h>

const int sensorPin = A0;
const unsigned long meetTijd = 60000;

float TempC = 0.0;
float a = 639.5, b = -0.1332, c = -162.5;

float Read_NTC10k(int pin) {
  int raw = analogRead(pin);
  if (raw == 0) return -273.15;
  float Vntc = (raw * 3.3) / 4095.0;
  float Rntc = 10000.0 * ((3.3 / Vntc) - 1.0);
  TempC = a * pow(Rntc, b) + c;
  return TempC;
}

void setup() {
  Serial.begin(115200);
  analogReadResolution(12);
  Serial.println("--- UITGEBREIDE RUIS & MARGE ANALYSE ---");
}

void loop() {
  Serial.println("Meting loopt (60 sec)...");

  unsigned long startMillis = millis();
  unsigned long aantalMetingen = 0;

  double somTemp = 0;
  double somKwadratenTemp = 0;
  float maxTemp = -100.0;
  float minTemp = 200.0;

  while (millis() - startMillis < meetTijd) {
    float huidigeTemp = Read_NTC10k(sensorPin);

    somTemp += huidigeTemp;
    somKwadratenTemp += (double)huidigeTemp * (double)huidigeTemp;

    if (huidigeTemp > maxTemp) maxTemp = huidigeTemp;
    if (huidigeTemp < minTemp) minTemp = huidigeTemp;

    aantalMetingen++;
  }

  // BEREKENINGEN
  double gemiddelde = somTemp / aantalMetingen;
```

```
  double variantie = (somKwadratenTemp / aantalMetingen) - (gemiddelde *
gemiddelde);
  double ruisRMS = sqrt(variantie);
  float peakToPeak = maxTemp - minTemp;

  double zScore = 0.5 / ruisRMS;
  double percentageBinnenMarge = erf(zScore / sqrt(2.0)) * 100.0;

  // OUTPUT
  Serial.println("\n====================================");
  Serial.print("Totaal aantal metingen: "); Serial.println(aantalMetingen);
  Serial.print("Gemiddelde Temperatuur: "); Serial.print(gemiddelde, 3);
Serial.println(" °C");
  Serial.println("------------------------------------");
  Serial.print("Peak-to-Peak verschil:  "); Serial.print(peakToPeak, 3);
Serial.println(" °C");
  Serial.print("Standaarddeviatie (σ): "); Serial.print(ruisRMS, 4);
Serial.println(" °C");
  Serial.println("------------------------------------");
  Serial.print("Percentage metingen binnen\n1.0°C bereik (±0.5°C):  ");
  Serial.print(percentageBinnenMarge, 2); Serial.println(" %");
  Serial.println("====================================");

  if (percentageBinnenMarge > 95) {
    Serial.println("RESULTAAT: Betrouwbare sensor voor 1°C nauwkeurigheid.");
  } else {
    Serial.println("RESULTAAT: Te veel ruis voor 1°C nauwkeurigheid.");
  }

  Serial.println("\nHerstart over 5 seconden...");
  delay(5000);
}
```

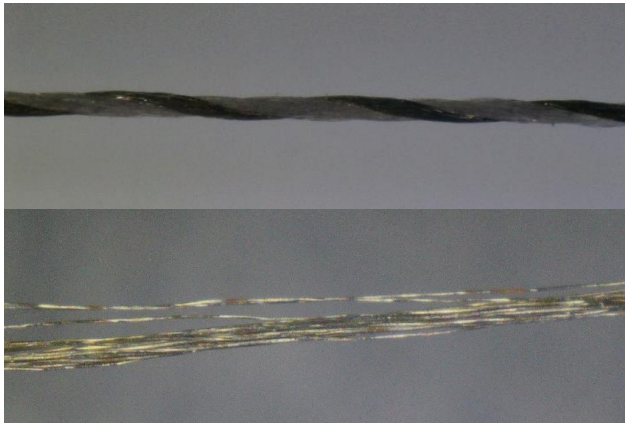
7 Conductive yarn as a temperature sensor

7.1 Measurements

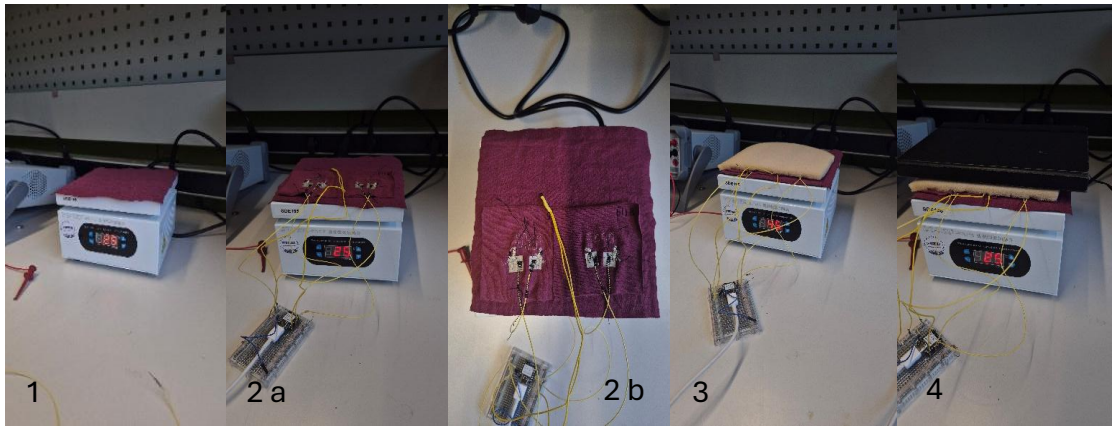
From the literature review, conductive yarns emerged as a promising approach for achieving non-invasive temperature monitoring in textiles. To assess whether the conductive yarns available at TU Delft are sufficiently sensitive to temperature changes, a series of preliminary measurements was conducted prior to this project. Based on these measurements, two yarns were selected for further development: Shieldex 78f20 and Silver-Tech RNNT7 (Appendix figure 5). These yarns showed the steepest resistance-temperature slope and the most linear response.

Sample preparation

Samples 1 and 2 were made using Shieldex 78f20. The yarn was put on a bobbin in the sewing machine and stitched on a piece of t-shirt fabric (90% cotton, 5% elastane, 5% viscose) in a pattern that can be found in (Appendix figure 7). These initial samples served as test samples to optimise the electrical connections, the data acquisition workflow, and the mechanical setup. Once the setup worked, all subsequent measurements were performed using this optimised configuration, this configuration can be found on the next page. After these samples, 3 to 8 were produced and tested in pairs. Samples 3 and 4 were sewn, samples 5 and 6 were twisted and sewn, and samples 7 and 8 were embroidered.



Appendix figure 5: Selected yarns, up: Silver-Tech RNNT7, down: Shieldex 78f20



Appendix figure 6: Set-up of measurements

Set-up of measurements

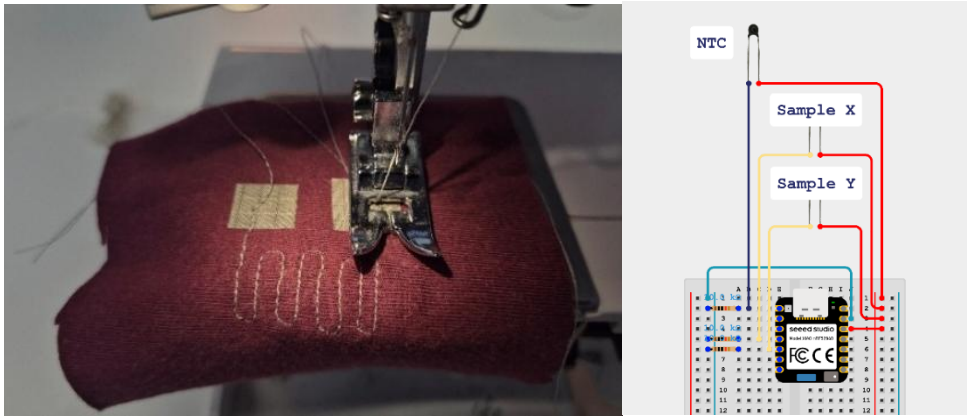
The measurement set-up consisted of the following layers, from bottom to top (Appendix figure 6):

6. T-shirt fabric as the bottom layer.
7. Conductive yarn samples and a reference NTC thermistor, see detailed in Appendix figure 6 2b.
8. Foam layer to ensure even pressure and good thermal contact.
9. A weight (notebook) is placed on top to maintain stable contact with the heating plate.

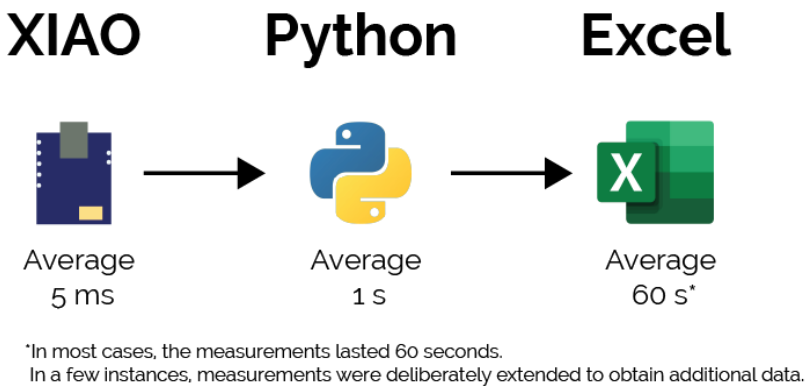
This configuration ensured consistent heat transfer and reliable resistance readings throughout all measurements.

Connection to the XIAO

Each yarn sample was stitched onto a piece of T-shirt fabric. Two small patches of Shieldit Super (an iron-on conductive fabric) were applied to the fabric to serve as contact pads. The conductive yarns were stitched across these pads, and two insulated copper wires connected the pads to a breadboard. The breadboard housed the Seeed XIAO microcontroller and three 10 kΩ resistors used for resistance measurement and data acquisition (Appendix figure 7, Appendix figure 8). The choice of a 10 kΩ resistor aligns with the initial characterisation of the yarn, where a 0.5 m sample demonstrated a resistance range of around 4 kΩ to 10 kΩ. The corresponding code is provided in Appendix 7.2



Appendix figure 7: Connection to XIAO, left: Sewing over Shieldit Super, right: Schematic of XIAO



Appendix figure 8: Data acquisition system (in most cases)

Measuring samples

The Arduino code output the average resistance every 5 ms, which was streamed to Python. Python then calculated a 1-second averaged value to reduce noise. For most test conditions, the analysis was based on 60 seconds of averaged data (Appendix figure 8).

Before recording, the system was allowed to stabilise once the NTC indicated that the hotplate had reached the target temperature; an additional 5-minute waiting period was used to ensure thermal stability and improve reliability.

Samples and measurements

After the initial optimisation phase, six additional samples were prepared and measured, with two samples tested simultaneously at each measurement. In Table 6 and Table 7 a summary of the sample characteristics and the corresponding measurement procedures. Pictures of these samples are shown in (Appendix figure 10).

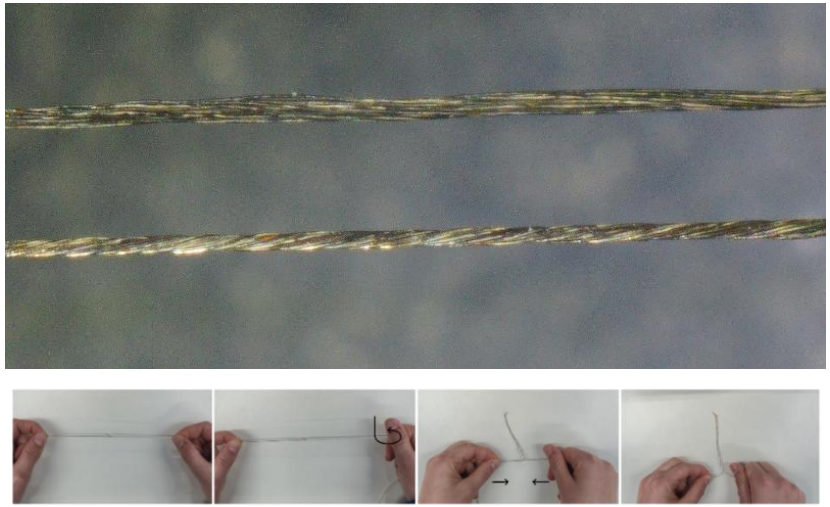
Table 6: Specifications of the tested yarn samples, categorised by structure and material.

Sample ID	Processing	Material	Description / Structure	Length (mm)
Sample 3	Sown	Shieldex 78f20	Standard conductive yarn	119
Sample 4	Sown	Silver-Tech RNNT7	Standard conductive yarn	135
Sample 5	Twisted and sown	Shieldex 78f20	Lightly twisted and coiled*	106
Sample 6	Twisted and sown	Shieldex 78f20	Strongly twisted and coiled*	112
Sample 7	Embroidery	Shieldex 78f20	Smooth, flowing motif	-
Sample 8	Embroidery	Shieldex 78f20	Angular, cornered motif	-

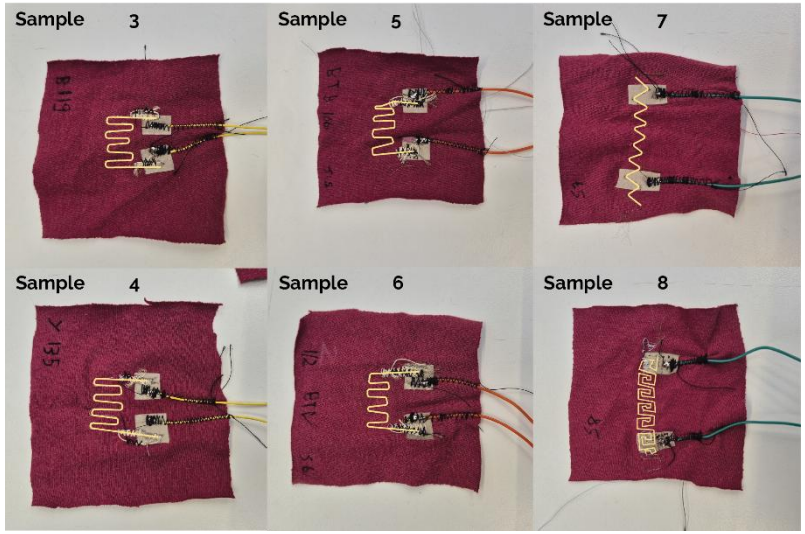
**Note: Based on multimeter tests, the yarn was twisted and coiled (Appendix figure 10), suggesting that twisting reduced resistance and possibly improved signal stability*

Table 7: Description of thermal testing conditions applied to the different sample groups

Applicable Samples	Measurement ID	Type	Description / Procedure	Duration (s)
S3 and S4 (Sown)	M1	Static Range	Temp. sweep: 25 °C to 45 °C (5°C increments).	60
	M2	Sensitivity	Temp. sweep: 25 °C to 30 °C (1°C increments) to detect small-scale changes.	60
	M3	Dynamic	Step-response: Plate held at 30 °C (1 min), then increased by 1 °C steps triggered by NTC detection.	723
S5 and S6 (Twisted)	M1	Static Range	Temp. sweep: 20 °C to 45 °C (5°C increments).	60
	M2	Repeatability	Plate at 45 °C. Sample placed On / Off (2 cycles) to test contact sensitivity.	359
S7 and S8 (Embroidery)	M1	Static Range	Temp. sweep: 20 °C to 45 °C (5°C increments).	60
	M2	Repeatability	Plate at 45 °C. Sample placed On / Off (1 cycle).	385



Appendix figure 10: Twisted and coiled yarn, up: Shieldex normal vs twisted, down: how Shieldex was twisted and coiled, creating a double yarn



Appendix figure 9: Samples

Results

Measurement 1 - Samples 3, 4, 5, 6, 7 and 8

All corresponding graphs are included in Appendix 7.4. Below is a summary of the results from measurements 1, 2 and 3. Because the results across samples follow similar patterns, the outcomes of measurement 1 are presented in a comparative table (Table 8). The only major difference is that samples 3 and 4 were tested starting at 25-45 °C, whereas all subsequent samples were measured from 20-45 °C.

Table 8: Measurement 1 of all samples

M1	Slope (dR/dT) (Ω/°C)	Linearity	Noise at 25 °C (Ω)	Resistance at 25 °C (Ω)
S3	7.35	Very	7.6	674.98
S4 (Silver tech)	0.35	Not so linear	5.73	224.04
S5	0.67	Linear	4.41	252.51
S6	0.49	A bit linear	4.31	272.25
S7	2.06	Linear	2.37	439.51
S8	2.62	A bit linear	2.4	614.89

From measurement 1 of samples 3 and 4, it became clear that the Silver-Tech yarn performs significantly worse than the Shieldex yarn. Silver-Tech showed a very low slope and poor linearity, making it unsuitable for accurate temperature sensing. Therefore, only the Shieldex 78f20 yarn was used in the other measurements.

Measurement 2 - Samples 3 and 4

The results of measurement 2 are displayed in Table 9. In Table 10 a comparison is shown of measuremenst 1 and 2 of sample 3.

Table 9: Measurement 2 - Samples 3 and 4

M2	Slope (dR/dT) (Ω/°C)	Linearity	Noise at 25 °C (Ω)	Resistance at 25 °C (Ω)
S3	14.62	Very	2.95	666.49
S4	-	Not Linear	3.8	215.63

Table 10: Comparison of resistance at 30 °C - Sample 3

M1 S3 resistance at 30 °C (Ω)	M2 S3 resistance at 30 °C (Ω)
640.88	593.391

Measurement 3 - Samples 3 and 4

Sample 3: The resistance gradually decreases as the NTC temperature increases, consistent with earlier observations.

Sample 4: No measurable change in resistance was observed despite increasing NTC readings. The yarn appears insensitive to these small temperature steps.

Measurement 2 - Samples 5 and 6

Sample 5: The resistance is higher at lower temperatures and lower at higher temperatures, which aligns with expectations. However, the measurements do not follow the smooth thermal curve of the NTC, multiple abrupt fluctuations occur, suggesting instability or contact inconsistencies.

Sample 6: No relationship between temperature and resistance could be identified.

Measurement 2 - Samples 7 and 8

Sample 7 (flowing pattern): Resistance increases with increasing NTC temperature. This behaviour contradicts earlier findings (where resistance consistently decreased with rising temperature).

Sample 8 (angular pattern): No relationship between temperature and resistance was observed.

Discussion

Several factors could have affected the reliability of the measurements, contributing to the inconsistent resistance-temperature relationships observed across the samples.

- **Mechanical and material variability:** The methods used to process the yarn (stitching, twisting, embroidery) could add to mechanical stress, altering its internal structure and fibre alignment, which could have contributed to the inconsistent values.
- **Electrical instability:** Creating a stable electrical connection was a challenge. The transition from soft yarn to rigid wires (via stitching or breadboards) introduces variable contact resistance. Additionally, the Arduino's measurement system has picked up electrical noise, which obscured very small resistance changes.
- **Thermal inconsistencies:** The measurement setup consisted of multiple layers (foam, fabric, weights). This could prevent the heat from reaching the yarn evenly. If there were small air gaps between the yarn and the hotplate, the yarn would not warm at the same rate as the reference sensor, resulting in a data mismatch.
- **Circuit sensitivity limitations:** The resistor of 10 kΩ was disproportionately large compared to the actual resistance of the twisted and embroidered yarn samples, which dropped to a range of 200-600 Ω. In a voltage divider circuit, measurement sensitivity is maximised when the fixed resistor roughly matches the sensor's baseline resistance. This mismatch (10,000 Ω vs. ~400 Ω) compressed the output signal into a very narrow range. Consequently, small changes in resistance due to temperature fluctuations could result in small voltage shifts, which were likely masked by the quantisation error or electrical noise of the Arduino's ADC.

The inconsistent results were likely due to one of the following reasons. Alternatively, this specific yarn may not be suitable for measuring temperature.

Although Shieldex exhibits some temperature sensitivity, the variability among measurements is too large, and the temperature-resistance relationship is too unstable in this setup. This makes it difficult to achieve the required 1 °C precision, which is essential for this project.

Conclusion

Although Shieldex initially appeared promising with a slope of approximately 7.4 Ω/°C, further tests revealed considerable inconsistencies. The resistance at 30 °C differed by 47 Ω between measurement 1 and measurement 2, indicating that the same yarn does not exhibit the same increase or decrease in resistance at the same temperature. Additionally, the slope varied substantially between measurement methods:

- **7.4 Ω/°C** with 5 °C steps
- **14.6 Ω/°C** with 1 °C steps

If the yarn was stable, these values should be consistent.

For the twisted samples (5 and 6), twisting reduced the range and the slope (0.5-0.7 Ω/°C), making precise temperature detection difficult. The embroidered samples (7 and 8) also showed no reliable correlation between temperature and resistance. Differences in linearity between identical yarns suggest that the geometry of the stitched pattern affects behaviour, reducing reproducibility.

Measurement 3 (samples 3 and 4) confirmed that Shieldex responds to temperature changes, but the response is neither consistent nor predictable. The second measurements of samples 5-8 showed fluctuations in resistance, but these were irregular and not aligned with the smooth temperature progression indicated by the NTC sensor.

Across measurements, Shieldex was shown to be inconsistent and unpredictable:

- Slope values differed between measurements.
- Identical samples produced different resistances at the same temperature.
- Twisting or embroidery altered the electrical behaviour significantly.
- None of the samples had a resistance range small enough to detect 1 °C changes reliably.

For accurate temperature sensing, the range should be smaller than the slope, but this was not the case for any of the tested samples. As a result, none of the yarns demonstrated the stability or accuracy required for this application.

These results indicate that these conductive yarns are unsuitable for use as primary sensors in this temperature-monitoring setup. Because the material is so unpredictable, the NTC sensor is a better, more reliable fit for now, to integrate into the proof-of-concept.

This does not mean that the idea of a textile sensor should be discarded. It remains a promising direction for developing a non-invasive smart wearable. If further research is conducted, for example, using alternative yarn types or improved connections, it may be possible to replace the NTC with conductive yarn in a future version of the wearable.

7.2 Arduino code used

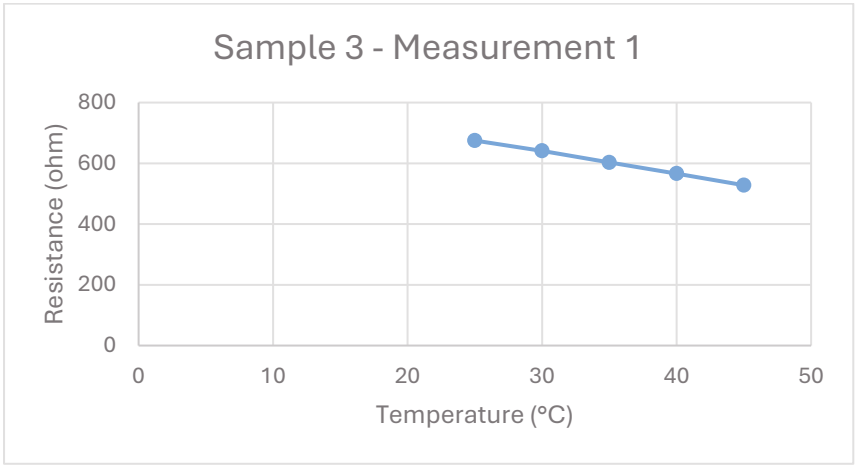
```
#include <math.h>

const int pinRx  = A3;          // Weerstand op A3
const int pinNTC = A0;          // NTC op A0
const int pinRz  = A4;          // Tweede weerstand op A4

const float Vcc  = 3.3;         // Voedingsspanning XIAO
const float Rref = 10000.0;     // Referentieweerstand 10k

void setup() {
  Serial.begin(115200);
  analogReadResolution(12);     // XIAO: 12-bit ADC (0..4095)
}

void loop() {
  unsigned long start = micros();
  const unsigned long duration = 5000; // 5 ms venster

  long sumRawRx  = 0;
  long sumRawNTC = 0;
  long sumRawRz  = 0;
  int count = 0;

  // Verzamel zoveel mogelijk samples in 5 ms
  while (micros() - start < duration) {
    int rawRx  = analogRead(pinRx);
    int rawNTC = analogRead(pinNTC);
    int rawRz  = analogRead(pinRz);

    sumRawRx  += rawRx;
    sumRawNTC += rawNTC;
    sumRawRz  += rawRz;
    count++;
  }

  // Gemiddelden berekenen
  float avgRawRx  = (float)sumRawRx  / count;
  float avgRawNTC = (float)sumRawNTC / count;
  float avgRawRz  = (float)sumRawRz  / count;

  // Rx berekenen (A3)
  float VoutRx = (avgRawRx / 4095.0f) * Vcc;
  float Rx     = Rref * (Vcc - VoutRx) / VoutRx;

  // Rz berekenen (A4)
  float VoutRz = (avgRawRz / 4095.0f) * Vcc;
  float Rz     = Rref * (Vcc - VoutRz) / VoutRz;

  // Temperatuur berekenen
  float Vntc = (avgRawNTC / 4095.0f) * Vcc;
  float Rntc = Rref * ((Vcc / Vntc) - 1.0f);
  float TempC = 639.5f * pow(Rntc, -0.1332f) - 162.5f;

  // Resultaat printen: TempC, Rx (A3), Rz (A4)
  Serial.print(TempC, 6);
  Serial.print(" ");
  Serial.print(Rx, 6);
  Serial.print(" ");
  Serial.println(Rz, 6);
}
```

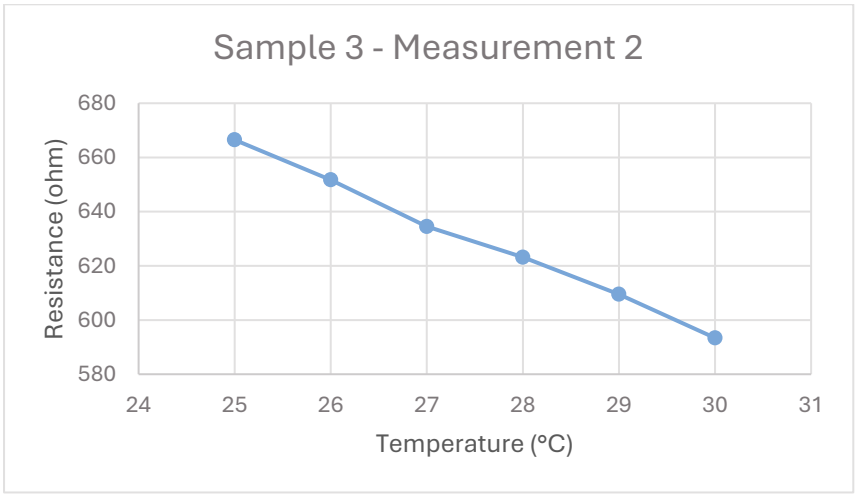
7.3 Python code used

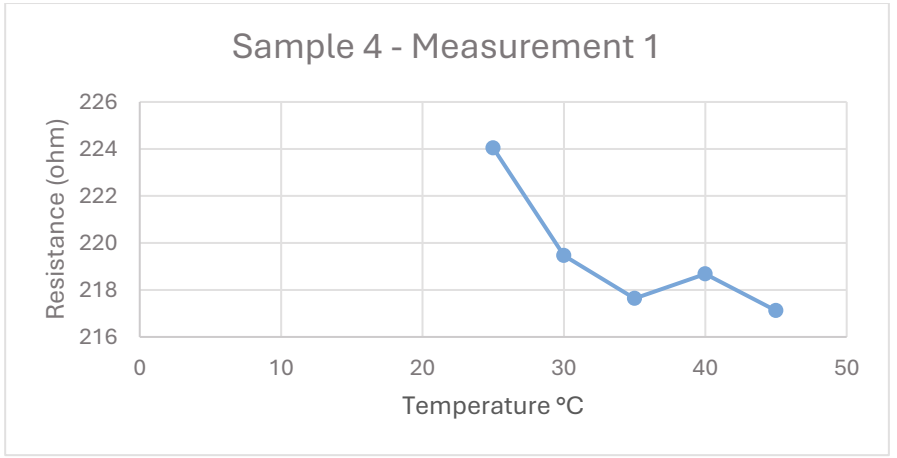
```
import serial, time, os, csv
from collections import defaultdict

PORT = 'COM8'
BAUDRATE = 115200
CSV_PATH = os.path.abspath("gemiddeld.csv")
OFFSET_NTC = -4.71  # kalibratieverschil in graden Celsius


# Labels per kolomblok (4 kolommen per run)
HEADERS = [
    "Tijd (s)",
    "Gemiddelde temperatuur (°C)",
    "Gemiddelde weerstand A3 (Ω)",
    "Gemiddelde weerstand A4 (Ω)"
]

def load_sheet(path):
    if not os.path.exists(path):
        return []
    with open(path, "r", newline="", encoding="utf-8") as f:
        return list(csv.reader(f, delimiter=';'))

def save_sheet(path, rows):
    with open(path, "w", newline="", encoding="utf-8") as f:
        writer = csv.writer(f, delimiter=';')
        writer.writerows(rows)

def count_runs(rows):
    # Aantal bestaande kolomblokken afleiden uit het aantal keer dat de eerste
header voorkomt
    if not rows:
        return 0
    header_row = rows[0]
    return sum(1 for cell in header_row if cell == HEADERS[0])

def ensure_size(rows, min_rows, min_cols):
    # Zorg dat er genoeg rijen/kolommen zijn
    while len(rows) < min_rows:
```

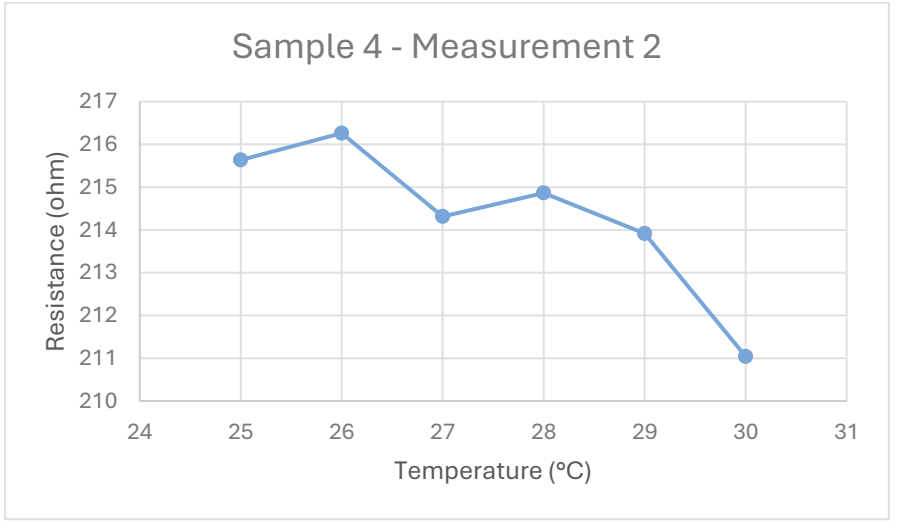
```
        rows.append([])
    for r in range(len(rows)):
        while len(rows[r]) < min_cols:
            rows[r].append("")

def main():
    rows = load_sheet(CSV_PATH)

    run_index = count_runs(rows)   # 0 voor eerste run, 1 voor tweede, etc.
    block_width = len(HEADERS)     # 4 kolommen per run
    offset = run_index * block_width

    # Plaats koptekst voor deze run
    ensure_size(rows, 1, offset + block_width)
    for i, h in enumerate(HEADERS):
        rows[0][offset + i] = h

    ser = serial.Serial(PORT, BAUDRATE)
    time.sleep(2)

    start = time.time()
    buffers = defaultdict(lambda: {"temp": [], "rx": [], "rz": []})
    last_written = -1

    try:
        while True:
            line = ser.readline().decode(errors='ignore').strip()
            if not line:
                continue

            parts = line.split()
            if len(parts) != 3:
                continue

            try:
                temp = float(parts[0]) + OFFSET_NTC
                rx   = float(parts[1])
                rz   = float(parts[2])
            except ValueError:
                continue

            sec = int(time.time() - start)
            buffers[sec]["temp"].append(temp)
            buffers[sec]["rx"].append(rx)
            buffers[sec]["rz"].append(rz)

            # Zodra we een nieuwe seconde ingaan, schrijf de vorige seconde weg
            if sec > last_written:
                if last_written in buffers:
                    temps = buffers[last_written]["temp"]
                    rxs   = buffers[last_written]["rx"]
                    rzs   = buffers[last_written]["rz"]
                    if temps and rxs and rzs:
                        avg_temp = sum(temps) / len(temps)
                        avg_rx   = sum(rxs)   / len(rxs)
                        avg_rz   = sum(rzs)   / len(rzs)

                        row_idx = last_written + 1  # rij 0 = headers
                        ensure_size(rows, row_idx + 1, offset + block_width)

                        rows[row_idx][offset + 0] = str(last_written)
                        rows[row_idx][offset + 1] = f"{avg_temp:.2f}"
                        rows[row_idx][offset + 2] = f"{avg_rx:.2f}"
                        rows[row_idx][offset + 3] = f"{avg_rz:.2f}"

                        print(f"[{last_written}s] Temp: {avg_temp:.2f} °C  Rx(A3):
{avg_rx:.2f} Ω  Rz(A4): {avg_rz:.2f} Ω")

                last_written = sec

    except KeyboardInterrupt:
        ser.close()
        save_sheet(CSV_PATH, rows)
        print(f"Opgeslagen: {CSV_PATH}")

if __name__ == "__main__":
    main()
```

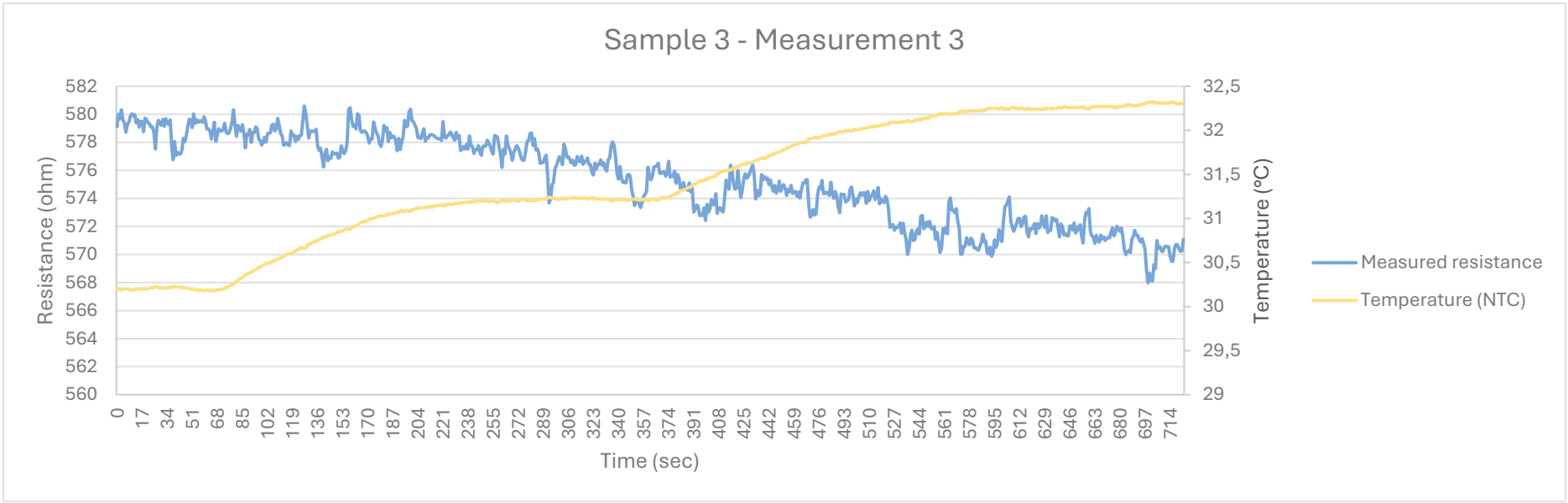
7.4 All corresponding graphs to the measurements

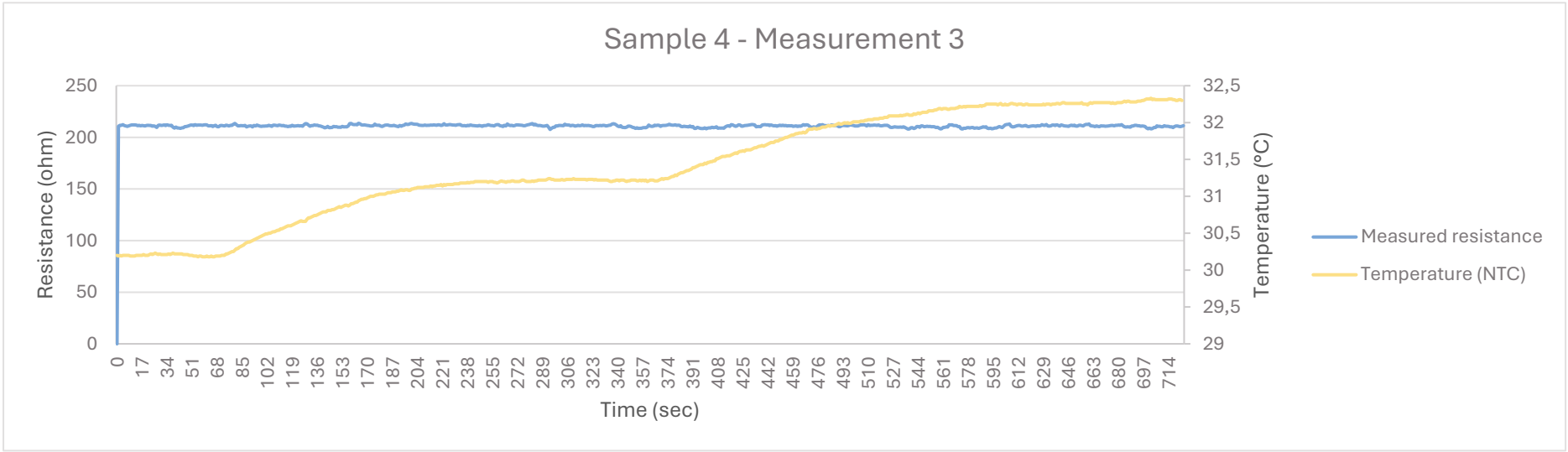


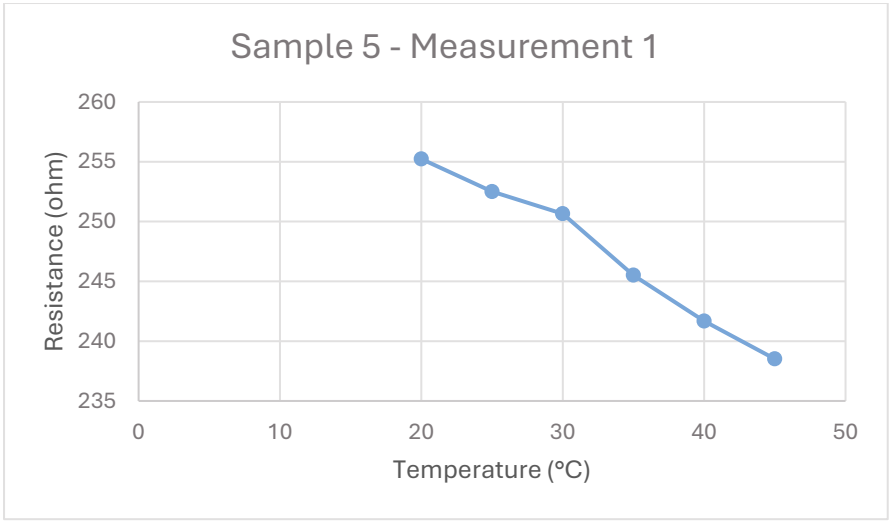


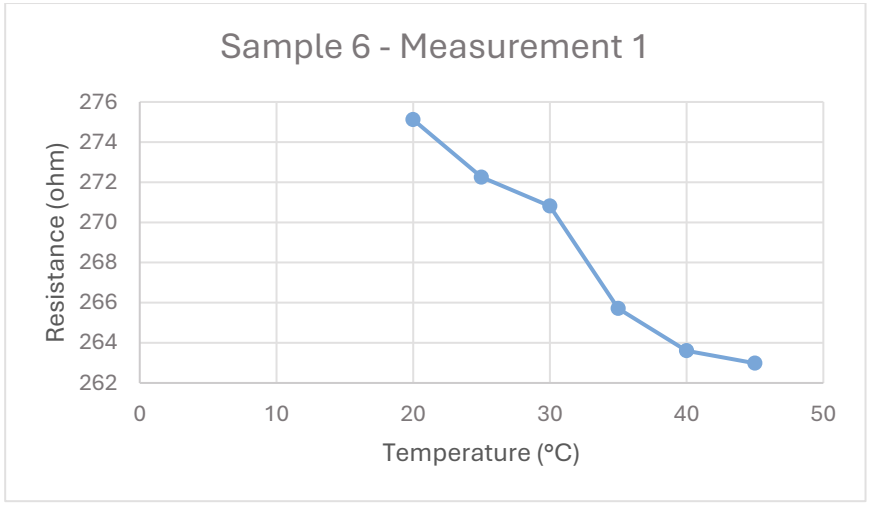


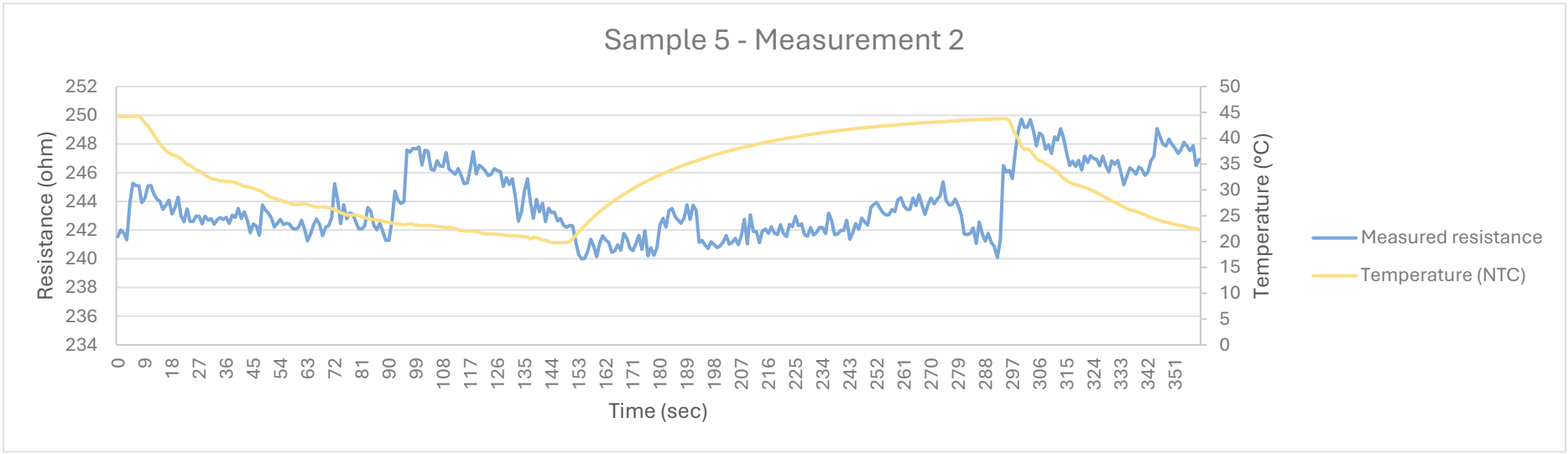


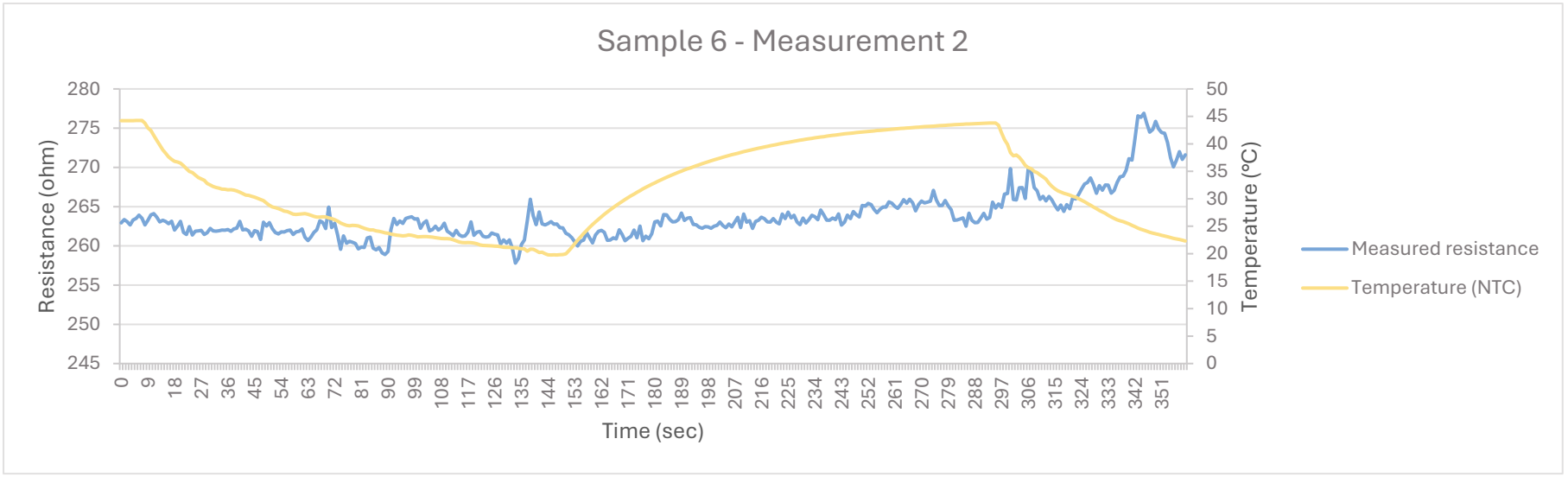

Sample 3 - Measurement 3
Resistance (ohm)
582
580
578
576
574
572
570
568
566
564
562
560
Temperature (°C)
32,5
32
31,5
31
30,5
30
29,5
29
0 17 34 51 68 85 102 119 136 153 170 187 204 221 238 255 272 289 306 323 340 357 374 391 408 425 442 459 476 493 510 527 544 561 578 595 612 629 646 663 680 697 714
Time (sec)
Measured resistance
Temperature (NTC)

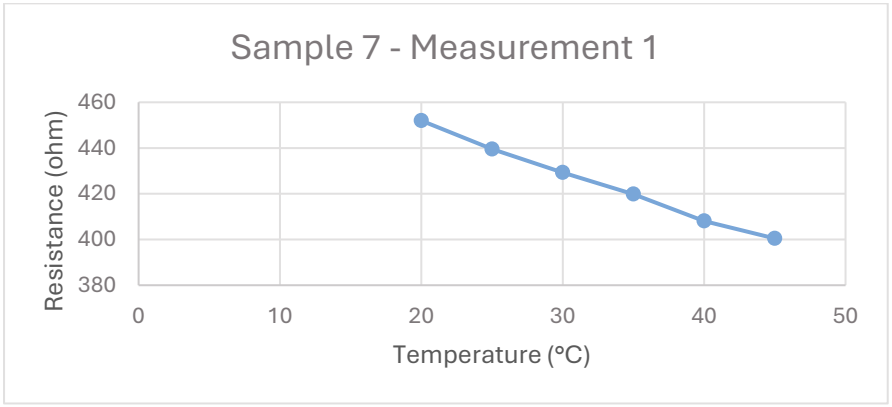

Sample 4 - Measurement 3
Resistance (ohm)
250
200
150
100
50
0
Temperature (°C)
32,5
32
31,5
31
30,5
30
29,5
29
0 17 34 51 68 85 102 119 136 153 170 187 204 221 238 255 272 289 306 323 340 357 374 391 408 425 442 459 476 493 510 527 544 561 578 595 612 629 646 663 680 697 714
Time (sec)
Measured resistance
Temperature (NTC)

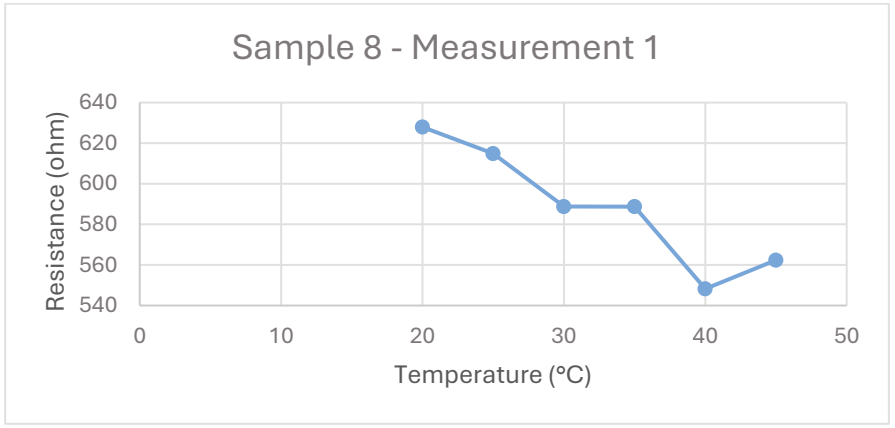

Sample 5 - Measurement 1
Resistance (ohm)
260
255
250
245
240
235
0
10
20
30
40
50
Temperature (°C)

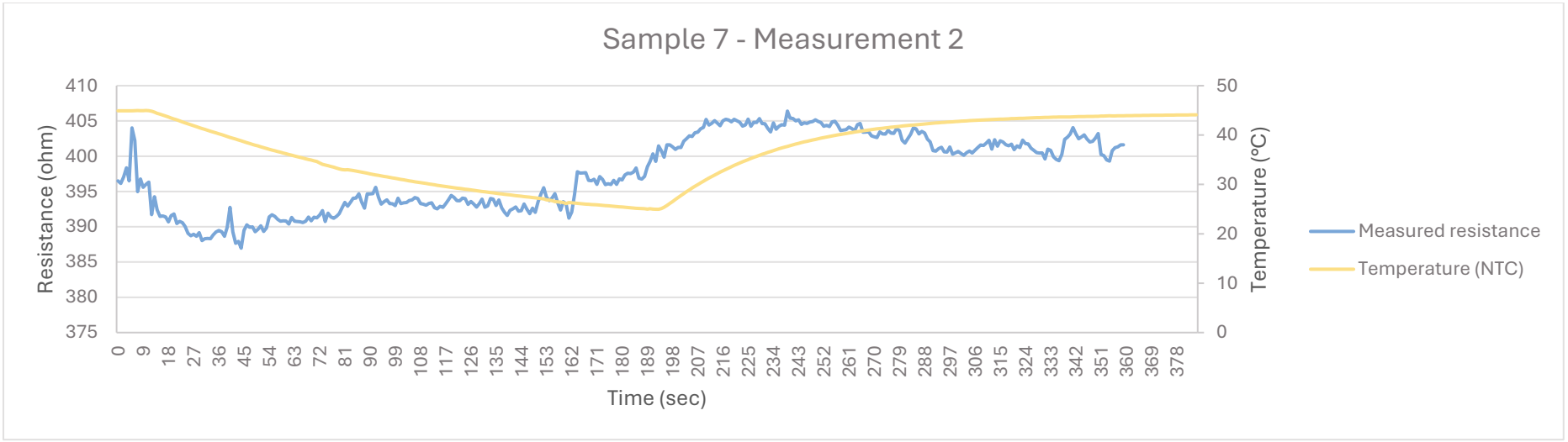

Sample 6 - Measurement 1
Resistance (ohm)
276
274
272
270
268
266
264
262
0
10
20
30
40
50
Temperature (°C)

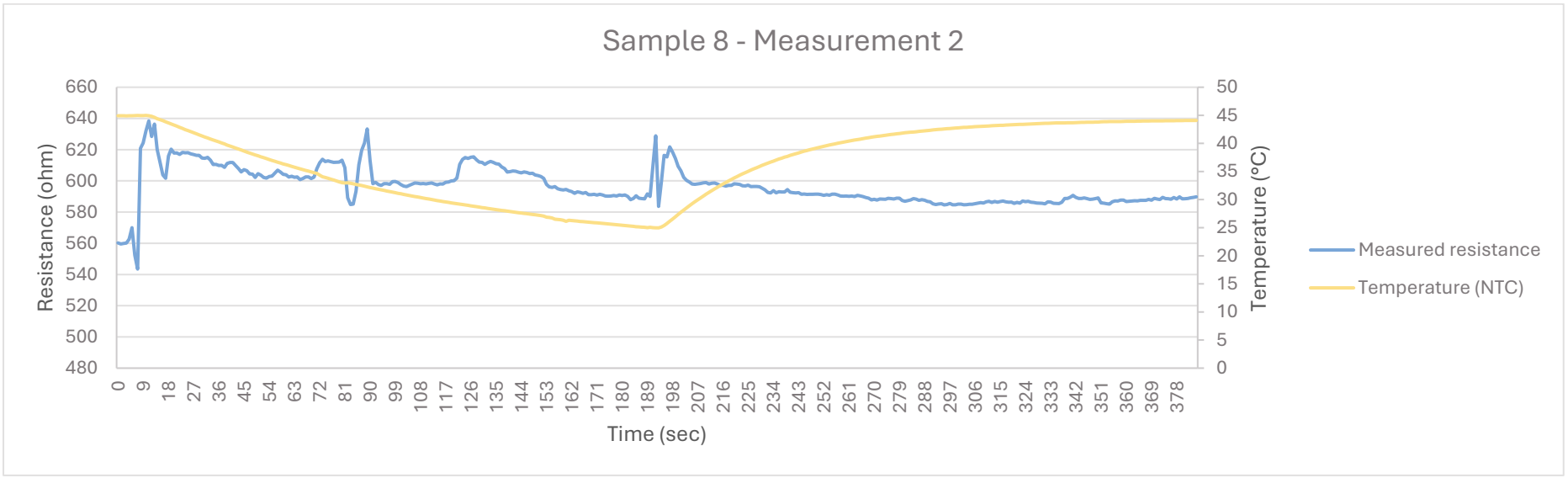

Sample 5 - Measurement 2
Resistance (ohm)
252
250
248
246
244
242
240
238
236
234
Temperature (°C)
50
45
40
35
30
25
20
15
10
5
0
0
9
18
27
36
45
54
63
72
81
90
99
108
117
126
135
144
153
162
171
180
189
198
207
216
225
234
243
252
261
270
279
288
297
306
315
324
333
342
351
Time (sec)
Measured resistance
Temperature (NTC)


Sample 6 - Measurement 2
Resistance (ohm)
Temperature (°C)
Time (sec)
Measured resistance
Temperature (NTC)


Sample 7 - Measurement 1
Resistance (ohm)
Temperature (°C)


Sample 8 - Measurement 1
Resistance (ohm)
Temperature (°C)


Sample 7 - Measurement 2
Resistance (ohm)
Temperature (°C)
Time (sec)
Measured resistance
Temperature (NTC)


Sample 8 - Measurement 2
Resistance (ohm)
Temperature (°C)
Time (sec)
Measured resistance
Temperature (NTC)

8 Acoustic methodology code

8.1 DJI microphone python code OA joint

```
import numpy as np
import soundfile as sf
import os
import matplotlib.pyplot as plt
from scipy.signal import butter, filtfilt, hilbert, find_peaks

# ========================================
# PAS HIER JE BESTANDSPAD AAN
# ========================================
# Zorg dat dit verwijst naar je DJI bestand
FILE_PATH =
r"C:\Users\Manon\Documents\Afstuderen\Afstuderen_Arduino\EPS\Geluidsfiles grafieken
voor verslag\DJI_11_20251114_142232.WAV"
# ========================================

def analyze_and_plot_knee(path):
    if not os.path.exists(path):
        print(f"⚠ Bestand niet gevonden op: {path}")
        return

    print(f"Bestand laden: {os.path.basename(path)}...")

    # 1. Load Audio (Met soundfile voor support)
    # sf.read geeft direct de audio als floats (-1.0 tot 1.0) en de samplerate (fs)
    try:
        audio, fs = sf.read(path)
    except Exception as e:
        print(f"❌ Fout bij laden bestand: {e}")
        return

    # Als het bestand Stereo is (2 kolommen), pakken we alleen Links
    if audio.ndim > 1:
        print(" -> Stereo bestand gedetecteerd, we gebruiken kanaal 1 (Links)")
        audio = audio[:, 0]

    # 2. Normalisation [-1.0, 1.0]
    # op de 'max piek' van dit specifieke bestand voor de analyse.
    max_val = np.max(np.abs(audio))
    if max_val > 0:
        audio /= max_val

    # 3. Bandpass Filtering (10 kHz - 20 kHz) - Toreyin et al.
    print("Filteren (10-20 kHz)...")
    low, high = 10000, 20000
    nyq = 0.5 * fs
    b, a = butter(N=6, Wn=[low/nyq, high/nyq], btype='band')
    filtered_signal = filtfilt(b, a, audio)

    # 4. Envelope Detection via Hilbert Transform
    print("Hilbert Envelope berekenen...")
    amplitude_envelope = np.abs(hilbert(filtered_signal))

    # 5. Adaptive Threshold (Mean + 4*STD)
    threshold = np.mean(amplitude_envelope) + 4.0 * np.std(amplitude_envelope)

    # 6. Peak Detection & Debouncing (10ms dead time)
    min_dist_samples = int(0.010 * fs)
    peaks, _ = find_peaks(amplitude_envelope, height=threshold,
distance=min_dist_samples)

    # ========================================
    # VISUALISATIE
    # ========================================
    plt.figure(figsize=(14, 6))
    t = np.arange(len(amplitude_envelope)) / fs

    # De Envelope plotten
    plt.plot(t, amplitude_envelope, label='Hilbert Envelope (10-20 kHz)',
color='#1d3557', lw=0.8)

    # De Threshold plotten
    plt.axhline(y=threshold, color='red', linestyle='--', alpha=0.7,
label=f'Threshold (Mean + 4*STD): {threshold:.4f}')

    # De gedetecteerde pieken markeren
    plt.plot(t[peaks], amplitude_envelope[peaks], "o", color='red',
label=f'Detected Crepitus (n={len(peaks)})')

    plt.title(f"Toreyin and Teague Analysis: Crepitus Detection, OA Joint",
fontsize=18)
    plt.xlabel("Time (s)", fontsize=16)
    plt.ylabel("Amplitude (Normalised)", fontsize=16)
    plt.legend(loc='upper right')
    plt.grid(alpha=0.2)
    plt.tight_layout()
    plt.show()

    print(f"☑ Analyse gereed. {len(peaks)} crepitaties gevonden.")

if __name__ == "__main__":
    analyze_and_plot_knee(FILE_PATH)
```

8.2 Code DJI comparison 3 files

```
import soundfile as sf
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.widgets import Slider, Button
import os
from scipy.signal import butter, filtfilt, resample, hilbert, find_peaks

# ========================================
# CONFIGURATION
# ========================================
FILES = {
    "Healthy": "DJI_29_20260128_155335-gezond.wav",
    "OA": "DJI_11_20251114_142232.wav",
    "Trousers": "DJI_32_20260128_161652-broek (1).wav"
}

ROW_LABELS = ["Healthy Joint (DJI)", "Osteoarthritis (OA) Joint (DJI)",
"Trousers/Clothing Noise (DJI)"]
TARGET_FS = 48000
GLOBAL_OFFSET = 0.5
OUTPUT_FOLDER = "grafiek tweaken DJI FILES"

if not os.path.exists(OUTPUT_FOLDER):
    os.makedirs(OUTPUT_FOLDER)

# ========================================
# SIGNAL PROCESSING FUNCTIONS
# ========================================

def butter_bandpass(lowcut, highcut, fs, order=6):
    nyq = 0.5 * fs
    low = max(0.01, lowcut / nyq)
    high = min(0.99, highcut / nyq)
    b, a = butter(order, [low, high], btype='band')
    return b, a

def get_hilbert_envelope(signal):
    return np.abs(hilbert(signal))

def load_audio_corrected(file_path, offset_sec=0):
    if not os.path.exists(file_path): return None, None
    audio, fs = sf.read(file_path)
    if len(audio.shape) > 1: audio = audio[:, 0]
    if fs != TARGET_FS:
        audio = resample(audio, int(len(audio) * TARGET_FS / fs))
        fs = TARGET_FS
    start_sample = int(offset_sec * fs)
    return audio[start_sample:], fs

# ========================================
# INTERACTIVE PLOTTING
```

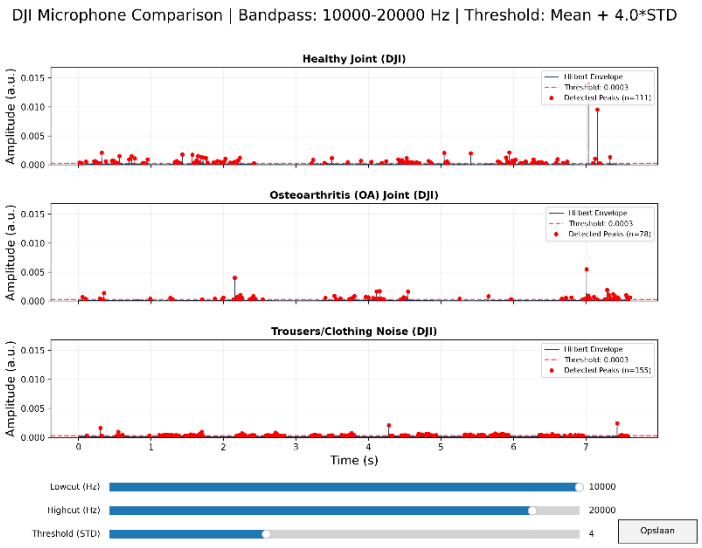
```
# ========================================

def interactive_tweaker():
    data_raw = {}
    available_lengths = []
    for key in FILES:
        audio, fs = load_audio_corrected(FILES[key], offset_sec=GLOBAL_OFFSET)
        if audio is not None:
            data_raw[key] = audio
            available_lengths.append(len(audio) / fs)

    max_time = min(available_lengths)
    num_samples = int(max_time * TARGET_FS)
    for key in data_raw:
        data_raw[key] = data_raw[key][:num_samples]

    fig, axes = plt.subplots(3, 1, figsize=(14, 10), sharex=True)
    plt.subplots_adjust(bottom=0.25, hspace=0.4)

    lines = []
    peak_dots = []
    threshold_lines = []
    legends = []

    t = np.arange(num_samples) / TARGET_FS
    for i in range(3):
        l, = axes[i].plot(t, np.zeros(num_samples), color='#1d3557', lw=0.7,
label='Hilbert Envelope')
        th = axes[i].axhline(0, color='red', linestyle='--', alpha=0.7,
label='Threshold')
        p, = axes[i].plot([], [], "o", color='red', markersize=4, label='Detected
Peaks')

        lines.append(l)
        peak_dots.append(p)
        threshold_lines.append(th)

        axes[i].set_title(ROW_LABELS[i], fontweight='bold')
        axes[i].set_ylabel("Amplitude (a.u.)", fontsize=14)
        axes[i].grid(True, alpha=0.2)
        # Maak een initiële legenda aan
        leg = axes[i].legend(loc='upper right', fontsize='small')
        legends.append(leg)

    ax_low = plt.axes([0.2, 0.15, 0.6, 0.03])
    ax_high = plt.axes([0.2, 0.11, 0.6, 0.03])
    ax_std = plt.axes([0.2, 0.07, 0.6, 0.03])
    ax_save = plt.axes([0.85, 0.07, 0.1, 0.04])
    axes[-1].set_xlabel("Time (s)",fontsize=14)


    s_low = Slider(ax_low, 'Lowcut (Hz)', 100, 10000, valinit=10000, valstep=100)
    s_high = Slider(ax_high, 'Highcut (Hz)', 2000, 22000, valinit=20000,
valstep=500)
```

```
    s_std = Slider(ax_std, 'Threshold (STD)', 1.0, 10.0, valinit=4.0, valstep=0.5)
    b_save = Button(ax_save, 'Opslaan')

    def update(val):
        low = s_low.val
        high = s_high.val
        std_m = s_std.val
        if low >= high: return

        b, a = butter_bandpass(low, high, TARGET_FS)
        oa_filt = filtfilt(b, a, data_raw["OA"])
        oa_env = get_hilbert_envelope(oa_filt)
        master_threshold = np.mean(oa_env) + std_m * np.std(oa_env)

        all_max = 0
        for i, key in enumerate(["Healthy", "OA", "Trousers"]):
            filt = filtfilt(b, a, data_raw[key])
            env = get_hilbert_envelope(filt)
            all_max = max(all_max, np.max(env))

            peaks, _ = find_peaks(env, height=master_threshold,
distance=int(0.01*TARGET_FS))

            lines[i].set_ydata(env)
            peak_dots[i].set_data(t[peaks], env[peaks])
            threshold_lines[i].set_ydata([master_threshold, master_threshold])

            # Update de legenda tekst met het aantal pieken
            legends[i].get_texts()[1].set_text(f'Threshold:
{master_threshold:.4f}')
            legends[i].get_texts()[2].set_text(f'Detected Peaks (n={len(peaks)})')

        for ax in axes:
            ax.set_ylim(0, all_max * 1.2)

        fig.suptitle(f"DJI Microphone Comparison | Bandpass: {int(low)}-{int(high)}
Hz | Threshold: Mean + {std_m}*STD", fontsize=18)
        fig.canvas.draw_idle()

    def save_plot(event):
        low, high, std_m = int(s_low.val), int(s_high.val), s_std.val
        filename = f"Crepitus_{low}-{high}Hz_plus{std_m}STD.png"
        path = os.path.join(OUTPUT_FOLDER, filename)
        # We verbergen de legenda niet, die hoort bij het plaatje
        plt.savefig(path, dpi=300)
        print(f"☑ Grafiek opgeslagen: {path}")

    s_low.on_changed(update)
    s_high.on_changed(update)
    s_std.on_changed(update)
    b_save.on_clicked(save_plot)

    update(None)
    plt.show()
```

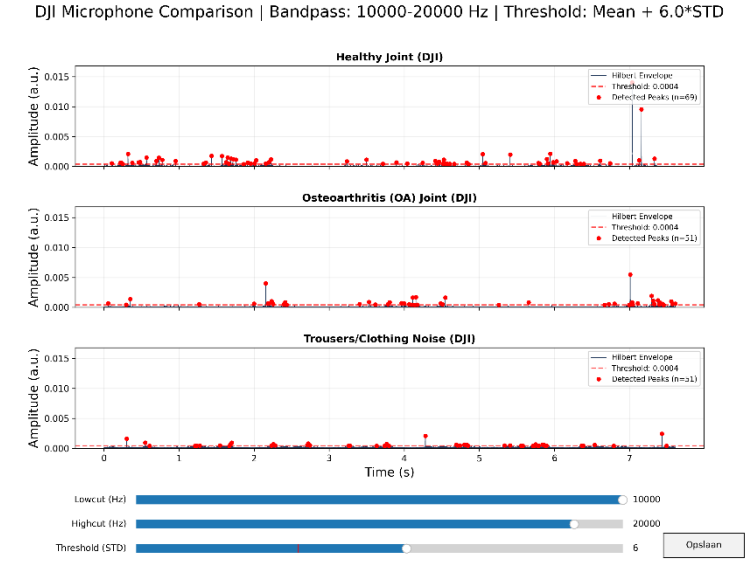
```
if __name__ == "__main__":
    interactive_tweaker()
```

8.3 Exploratory approach optimising crepitus detection.

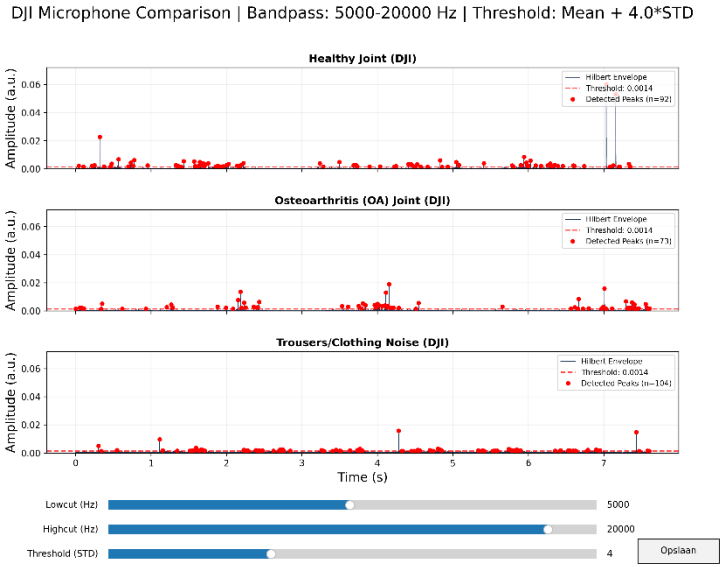
Trial 1



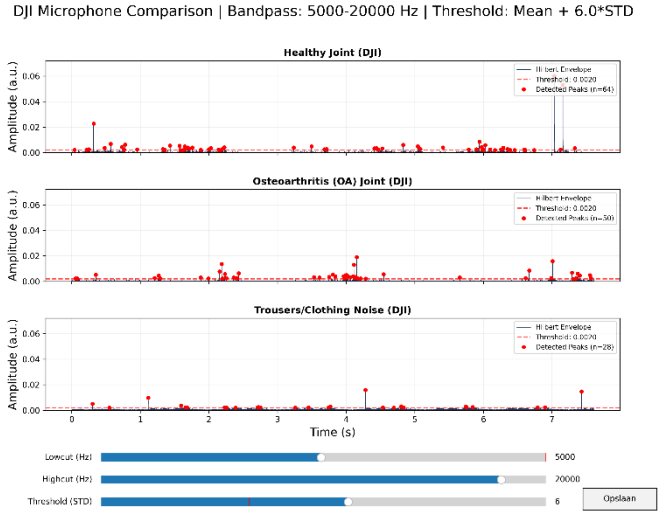
Trial 2



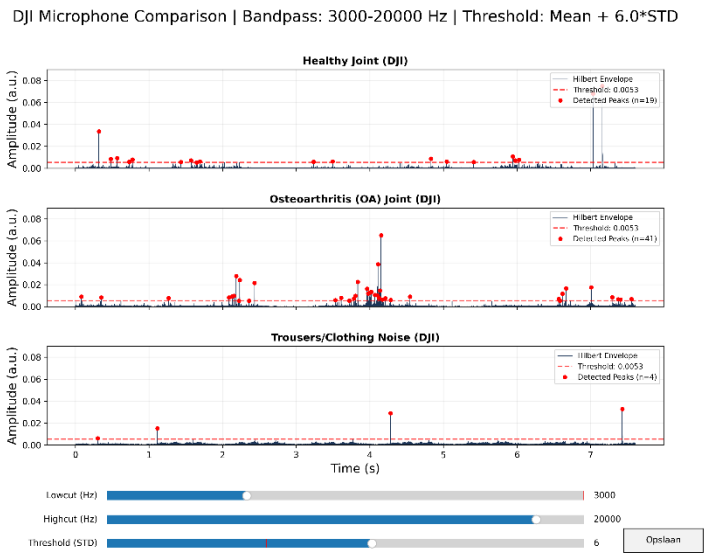
Trial 3



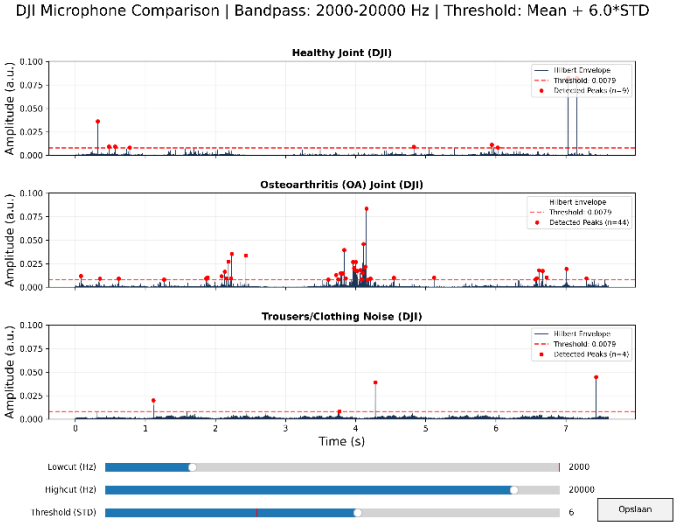
Trial 4



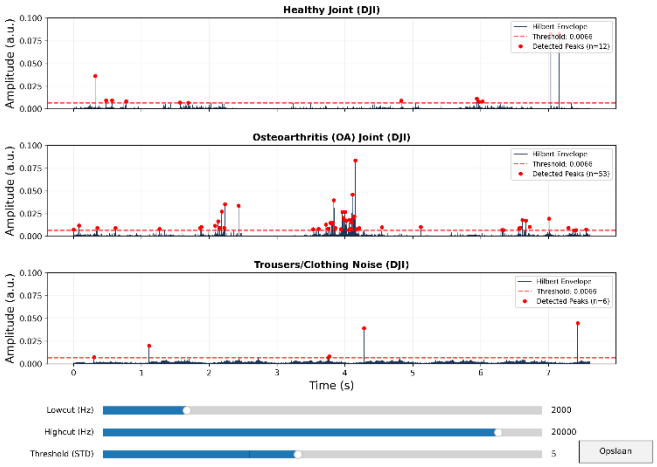
Trial 5



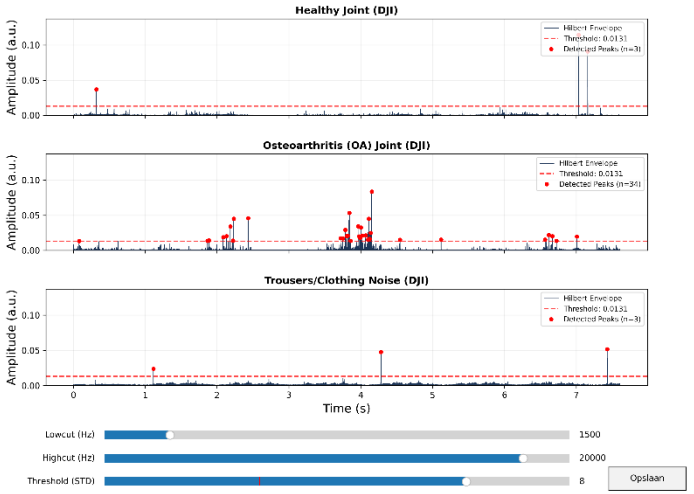
Trial 6



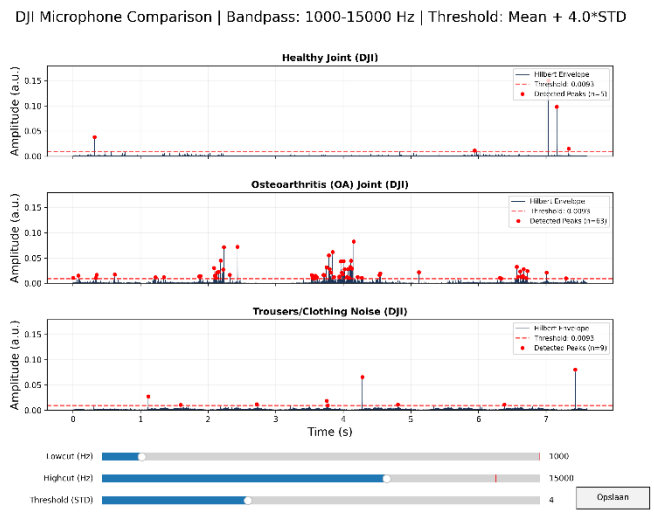
Trial 7



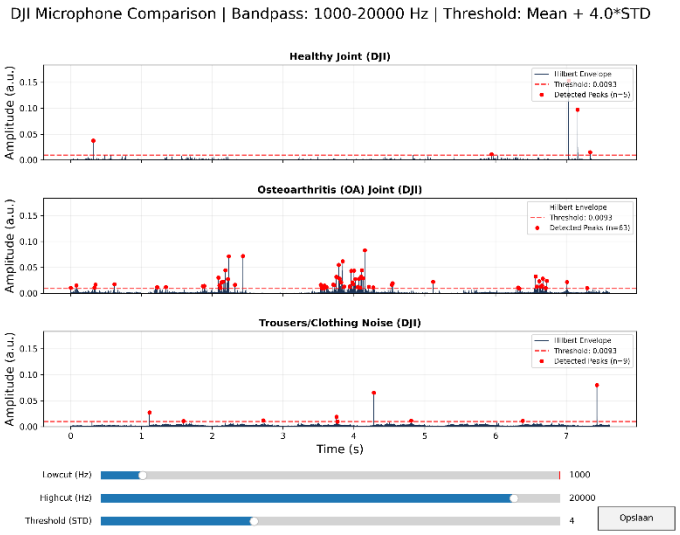
Trial 8



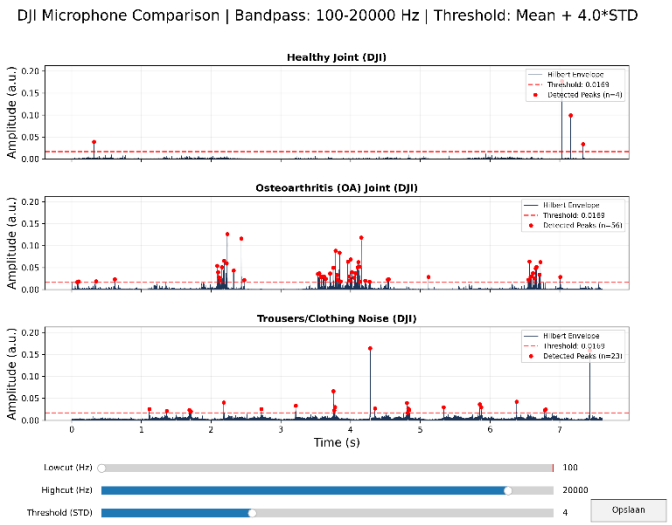
Trial 9



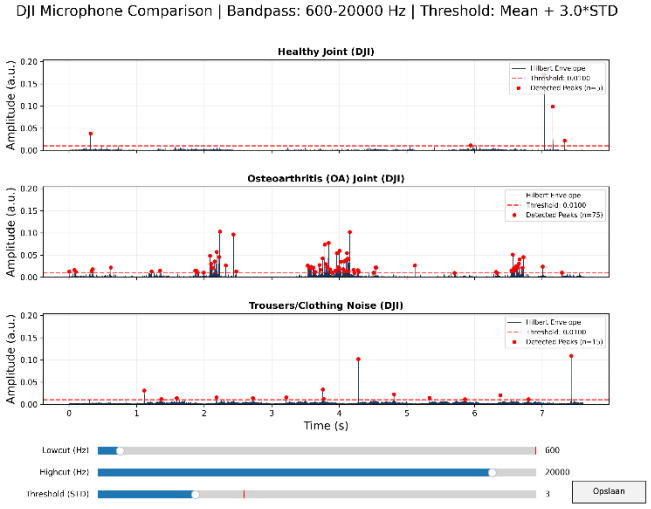
Trial 10



Trial 11

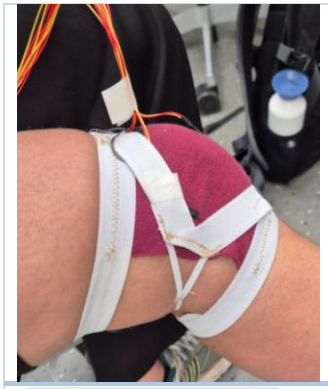
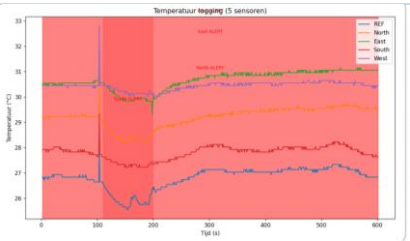
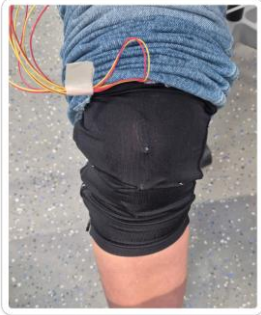
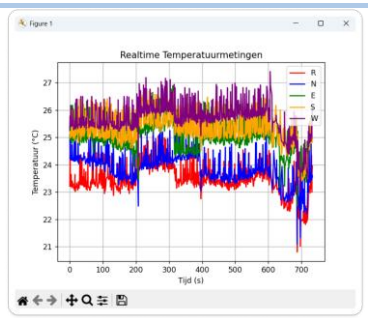
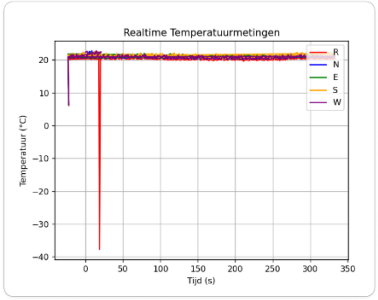
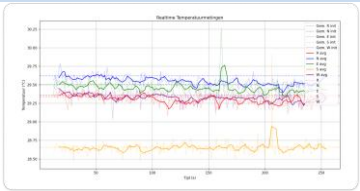
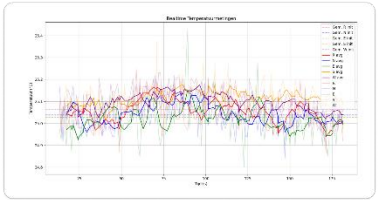


Trial 12



9 NTC tests

Table 11: Itterations of integration of NTCs

		While walking, the temperature sensors sensed a lower temperature. Indicating a good connection of the NTC's to the skin is important
		Smoothing the NTC data was necessary due to observed spikes.
		Short circuit.
	 	Calibrating the NTCs made the lines closer together.

10 Final Arduino code

```
#include <driver/i2s.h>
#include <SD.h>
#include <SPI.h>
#include "FS.h"
#include "esp_heap_caps.h"

// ==========================================
// CONFIGURATIE (PINNEN)
// ==========================================
#define I2S_WS       5   // D4
#define I2S_BCK      6   // D5
#define I2S_DATA     43  // D6
#define SD_CS        44  // D7 (Chip Select voor SD Kaart)

const int ntcPins[] = {A0, A1, A2, A3};

#define SAMPLE_RATE    48000
#define BUFFER_SAMPLES 1024

// We gebruiken een variabele bestandsnaam
char currentFilename[32] = "/storage0.bin";

// ==========================================
// DATA STRUCTUUR
// ==========================================
struct __attribute__((packed)) DataPacket {
    uint16_t syncWord;
    uint8_t packetType;
    uint32_t sequenceNumber;
    union {
        int16_t audio[BUFFER_SAMPLES];
        float temps[4];
    } payload;
};

QueueHandle_t dataQueue = NULL;
uint32_t globalSequenceCount = 0;

i2s_config_t i2s_config = {
    .mode = (i2s_mode_t)(I2S_MODE_MASTER | I2S_MODE_RX),
    .sample_rate = SAMPLE_RATE,
    .bits_per_sample = I2S_BITS_PER_SAMPLE_32BIT,
    .channel_format = I2S_CHANNEL_FMT_ONLY_LEFT,
    .communication_format = I2S_COMM_FORMAT_I2S,
    .intr_alloc_flags = ESP_INTR_FLAG_LEVEL1,
    .dma_buf_count = 8,
    .dma_buf_len = 512,
    .use_apll = false
};
```

```
// ==========================================
// NTC FUNCTIES
// ==========================================
float Read_Single_NTC(int pin) {
    float Vntc = (analogRead(pin) * 3.3) / 4095.0;
    if (Vntc < 0.05) return 0.0;
    float Rntc = 10000.0 * ((3.3 / Vntc) - 1.0);
    float a = 639.5, b = -0.1332, c = -162.5;
    return a * pow(Rntc, b) + c;
}

float Get_Avg_NTC(int pin) {
    float sum = 0;
    for(int i=0; i<10; i++) {
        sum += Read_Single_NTC(pin);
        delayMicroseconds(100);
    }
    return sum / 10.0;
}

// ==========================================
// BESTANDSNAAM KIEZEN
// ==========================================
void determineFilename() {
    int counter = 0;
    while (true) {
        snprintf(currentFilename, 32, "/storage%d.bin", counter);
        if (!SD.exists(currentFilename)) {
            Serial.print("🆕 Nieuw bestand aangemaakt: ");
            Serial.println(currentFilename);
            break;
        }
        counter++;
    }
}

// ==========================================
// TAAK 1: AUDIO (CORE 1)
// ==========================================
void audioTask(void *pvParameters) {
    static int32_t raw_buffer[BUFFER_SAMPLES];
    static DataPacket pkg;
    pkg.syncWord = 0xAAAA;

    while (true) {
        size_t bytes_read = 0;
        i2s_read(I2S_NUM_0, raw_buffer, sizeof(raw_buffer), &bytes_read,
portMAX_DELAY);

        if (bytes_read > 0) {
            pkg.packetType = 0; // 0 = Audio
            pkg.sequenceNumber = globalSequenceCount++;

            int samples_read = bytes_read / 4;
```

```
            for (int i = 0; i < samples_read; i++) {
                pkg.payload.audio[i] = (int16_t)(raw_buffer[i] >> 16);
            }
            // Stop in wachtrij voor opslag
            xQueueSend(dataQueue, &pkg, 0);
        }
    }
}
// ========================================
// TAAK 2: MANAGER & SD WRITER (CORE 0)
// ========================================
void dataManagerTask(void *pvParameters) {
    static DataPacket rx_pkg;
    static DataPacket temp_pkg;
    unsigned long lastTempMillis = 0;
    temp_pkg.syncWord = 0xAAAA;

    File dataFile;

    // Open het bestand één keer
    dataFile = SD.open(currentFilename, FILE_WRITE);
    if (!dataFile) {
        Serial.println("FOUT: Kan bestand niet openen!");
        while(1) { digitalWrite(LED_BUILTIN, HIGH); delay(100);
digitalWrite(LED_BUILTIN, LOW); delay(100); }
    }

    int packetsWrittenSinceSave = 0;

    while (true) {
        // --- A. TEMP METING ---
        if (millis() - lastTempMillis >= 1000) {
            lastTempMillis = millis();
            temp_pkg.packetType = 1; // 1 = Temp
            temp_pkg.sequenceNumber = globalSequenceCount;
            for(int i=0; i<4; i++) temp_pkg.payload.temps[i] =
Get_Avg_NTC(ntcPins[i]);

            // Stuur naar queue zodat hij OOK wordt opgeslagen op SD
            xQueueSend(dataQueue, &temp_pkg, 0);

            // Print status
            Serial.printf("REC: %s | Pkts: %d | Temp: %.1f\n",
                          currentFilename, globalSequenceCount,
temp_pkg.payload.temps[0]);
        }
        // --- B. SD SCHRIJVEN ---
        // Haal data uit de wachtrij
        if (xQueueReceive(dataQueue, &rx_pkg, pdMS_TO_TICKS(10))) {

            // Schrijf de ruwe bytes naar de SD kaart
            dataFile.write((uint8_t *)&rx_pkg, sizeof(DataPacket));
            packetsWrittenSinceSave++;

            // Elke 50 pakketjes (~1 keer per seconde) fysiek opslaan
            if (packetsWrittenSinceSave >= 50) {
                dataFile.flush();
                packetsWrittenSinceSave = 0;
                digitalWrite(LED_BUILTIN, !digitalRead(LED_BUILTIN)); // Knipper
LED bij save
            }
        }
    }
}
// ========================================
// SETUP
// ========================================
void setup() {
    Serial.begin(115200);
    analogReadResolution(12);
    pinMode(LED_BUILTIN, OUTPUT);
    delay(2000);

    Serial.println("\n--- START OFFLINE RECORDER ---");

    // Maak Queue in PSRAM
    dataQueue = xQueueCreateWithCaps(200, sizeof(DataPacket), MALLOC_CAP_SPIRAM);
    if (dataQueue == NULL) { Serial.println("FOUT: Geen PSRAM Queue!"); while(1); }

    // Start SD Kaart
    SPI.begin(7, 8, 9, SD_CS);
    if (!SD.begin(SD_CS)) {
        Serial.println("Geen SD kaart gevonden! Stop.");
        while(1);
    }
    Serial.println("SD Kaart OK.");

    // Bepaal bestandsnaam
    determineFilename();

    // I2S Starten
    i2s_driver_install(I2S_NUM_0, &i2s_config, 0, NULL);
    i2s_pin_config_t pins = { .bck_io_num = (gpio_num_t)I2S_BCK, .ws_io_num =
(gpio_num_t)I2S_WS, .data_out_num = I2S_PIN_NO_CHANGE, .data_in_num =
(gpio_num_t)I2S_DATA };
    i2s_set_pin(I2S_NUM_0, &pins);

    // Taken starten
    xTaskCreatePinnedToCore(audioTask, "Audio", 20000, NULL, 10, NULL, 1);
    xTaskCreatePinnedToCore(dataManagerTask, "SD_Writer", 20000, NULL, 5, NULL, 0);

    Serial.println(" Opname gestart! Koppel niet los zolang LED knippert.");
}

void loop() {
    // Loop is leeg, alles gebeurt in taken
    vTaskDelay(1000);
}
```

11 Python code .bin files reading

This code was part of the prototyping webpage, so it is not the whole streamlit architecture code, but only the part that shows how to unpack a .bin file

```
import streamlit as st
import struct
import wave
import csv
import os
import re
from datetime import datetime

# ========================================
# 1. Configuratie & Paden
# ========================================
# 2 (Sync) + 1 (Type) + 4 (Seq) + 2048 (Payload)
PACKET_SIZE = 2055
SYNC_WORD = 0xAAAA

# Paden bepalen
current_dir = os.path.dirname(os.path.abspath(__file__))
root_dir = os.path.dirname(current_dir)

PATH_TEMP = os.path.join(root_dir, "MetingenTemp")
PATH_SOUND = os.path.join(root_dir, "Sound", "Sound_bestanden")

os.makedirs(PATH_TEMP, exist_ok=True)
os.makedirs(PATH_SOUND, exist_ok=True)


# ========================================
# 2. Hulpfunctie: Bin Bestand Verwerken (MET RE-SYNC)
# ========================================
def process_bin_file(uploaded_file, output_base_name):
    """
    Zet een .bin bestand om naar .wav en .csv met Sync-Word detectie.
    """

    # Output paden
    wav_path = os.path.join(PATH_SOUND, f"{output_base_name}.wav")
    csv_path = os.path.join(PATH_TEMP, f"{output_base_name}.csv")

    # CSV openen
    csv_file = open(csv_path, "w", newline='')
    csv_writer = csv.writer(csv_file)
    csv_writer.writerow(["SequenceID", "Tijd_Geschat_sec", "T1", "T2", "T3", "T4"])

    # WAV openen
    wav_file = wave.open(wav_path, "wb")
    wav_file.setnchannels(1)
    wav_file.setsampwidth(2)
    wav_file.setframerate(48000)

    # We laden de bytes in het geheugen
    bytes_data = uploaded_file.getvalue()
    total_len = len(bytes_data)

    idx = 0
    packet_count = 0
    errors_fixed = 0

    try:
        # We lopen door de bytes heen (vergelijkbaar met while True in jouw script)
        while idx < total_len - 1: # Minimaal 2 bytes nodig voor sync check

            # 1. Zoek Sync Word (Byte 0 en 1)
            sync_bytes = bytes_data[idx : idx+2]
            current_sync = struct.unpack('<H', sync_bytes)[0]

            if current_sync != SYNC_WORD:
                # GEEN MATCH: Schuif 1 byte op en probeer opnieuw
                idx += 1
                errors_fixed += 1
                continue

            # 2. MATCH! Controleer of we de rest kunnen lezen
            # We hebben 2 bytes (sync) gehad, nu de rest (2053 bytes)
            payload_len = PACKET_SIZE - 2

            if idx + 2 + payload_len > total_len:
                break # Bestand stopt halverwege een pakketje

            # Lees de rest van het pakket
            # idx+2 slaan we over (dat was de sync), we lezen de rest
            rest_of_packet = bytes_data[idx+2 : idx+PACKET_SIZE]

            # Header decoderen (Type [1 byte] + Seq [4 bytes])
            # Dit zit in de eerste 5 bytes van 'rest_of_packet'
            header = struct.unpack('<BI', rest_of_packet[:5])
            pkt_type = header[0]
            seq_num = header[1]
            payload = rest_of_packet[5:] # De rest is audio of temp data

            if pkt_type == 0:
                # Audio
                wav_file.writeframes(payload)

            elif pkt_type == 1:
                # Temp (4 floats = 16 bytes)
                temp_bytes = payload[:16]
                temps = struct.unpack('<ffff', temp_bytes)
                time_est = seq_num * (1024 / 48000)
```

```
                # Filter gekke waarden
                if -50 < temps[0] < 150:
                    csv_writer.writerow([seq_num, f"{time_est:.4f}", temps[0],
temps[1], temps[2], temps[3]])

            packet_count += 1

            # Verspring met de volledige pakketgrootte naar het volgende blok
            idx += PACKET_SIZE

        return True, f"Succes! {packet_count} pakketjes. {errors_fixed} fouten
hersteld."

    except Exception as e:
        return False, str(e)
    finally:
        wav_file.close()
        csv_file.close()


# ==========================================
# 3. Hoofdfunctie UI
# ==========================================
def render_import():
    st.title("📂 SD Kaart Import")
    st.markdown("Sleep je `storageX.bin` bestanden hierheen. Het script zoekt
automatisch naar het sync-signaal (0xAAAA).")

    # 1. Datum kiezen
    selected_date = st.date_input("📅 Datum van meting", value=datetime.today())
    date_str = selected_date.strftime("%Y-%m-%d")

    # 2. File Uploader
    uploaded_files = st.file_uploader(
        "Sleep bestanden hierheen",
        type=["bin"],
        accept_multiple_files=True
    )

    # 3. Startknop
    if uploaded_files:
        st.info(f"{len(uploaded_files)} bestanden geselecteerd.")

        if st.button("⚙️ Verwerken starten", type="primary"):
            progress_bar = st.progress(0)

            for i, uploaded_file in enumerate(uploaded_files):
                original_name = uploaded_file.name

                # Haal nummer uit bestandsnaam (storage8.bin -> 8)
                match = re.search(r"storage(\d+)\.bin", original_name,
re.IGNORECASE)
```

```
                if match:
                    measure_num = match.group(1)
                    new_base_name = f"{date_str}_M{measure_num}"

                    with st.spinner(f"Bezig met {original_name}..."):
                        success, msg = process_bin_file(uploaded_file,
new_base_name)

                    if success:
                        st.success(f"☑ {new_base_name}: {msg}")
                    else:
                        st.error(f"❌ Fout {original_name}: {msg}")
                else:
                    st.warning(f"⚠ {original_name} overgeslagen (Naam moet
storageX.bin zijn)")

                progress_bar.progress((i + 1) / len(uploaded_files))
```

12 Validation acoustic sensors Works-like prototype

12.1 Test setup

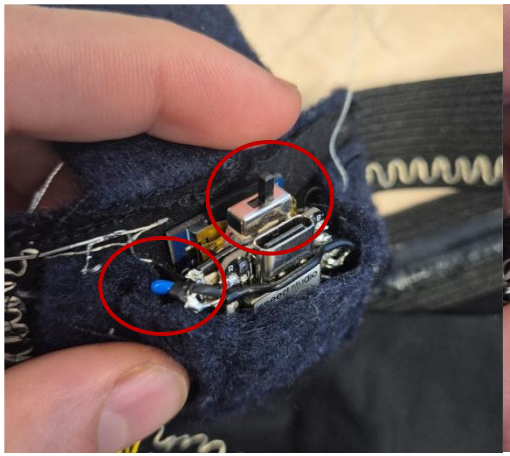
Deel 1: Instructies voor de meting

Wanneer meten? Ervaar je op dit moment meer gekraak of last dan normaal? Start dan een meting van ongeveer 10 tot 20 minuten. (Als het mogelijk is doe ook een meting wanneer je geen last hebt)

Bij vragen of problemen: xxx

Stappenplan:

1. **Aandoen:** Doe de wearable om zoals op de foto rechts te zien is.
2. **Inschakelen:** Zet de schakelaar (achter de USB-C-poort) om.
    - *Zet hem naar de kant van de blauwe stip = AAN.*



Stap 2: uit

Stap 2: aan

    - *Wacht 5 seconden voordat het oranje/gele lampje aangaat.*
3. **Check:** Controleer het **oranje/gele** LED-lampje. (wacht 5 seconden na het omschakelen)
    - **Rustig knipperen:** Alles is goed, de meting loopt. (Het oranje/gele lichtje is langer aan dan uit)
    - **Snel flikkeren:** Er is een fout (SD-kaart ontbreekt/zit niet goed).
        - *Gaat aan: begin dan met de oefeningen. Gaat hij niet aan ga dan door naar stap 4.*

Stap 1

4. **Aansluiten:** Steek de USB-C-kabel in de USB-C-poort.
    - *Let op: Houd bij het inpluggen de elektronica (de chip) goed tegen, zodat je niet te hard duwt.*
5. **Start:** Voer nu de oefeningen uit zoals beschreven.

Deel 2: Oefeningen

Benodigdheden:

- Een stevige stoel
- Loopruimte (bijv. rondom de eettafel)
- Een trap

Belangrijk: Audio-opname & Annotatie De wearable neemt continu geluid op. Dit betekent dat gesprekken hoorbaar zijn bij de analyse. Dit gebruik ik in het voordeel: **vertel hardop wat je doet.** Omdat ik niet kan *zien* wat je doet, is het handig dat ik het kan *horen*. Zeg bijvoorbeeld duidelijk: *"Ik begin nu met oefening 1"* of *"Ik neem nu 1 minuut rust"* of *"Ik heb nu geen last van mijn knie"*

1. **Start & Rust**

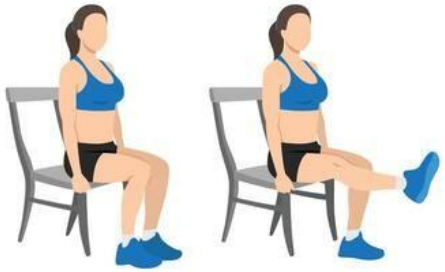
- Zet de wearable aan (zoals in stap 1 beschreven).
- Ga rustig zitten of staan en **wacht 2 minuten** voordat je begint.

2. **Oefening 1: Leg Extensions (zittend) (Appendix figure 11)**

- Zeg hardop: "Start Oefening 1" 1 keer aan het begin
- Ga op de stoel zitten. Strek je been volledig uit en buig weer terug.
- Herhaal dit **10 keer**.
- Rust: Wacht 1 minuut. Zeg hardop: "1 minuut rust”

3. **Oefening 2: Sit-to-Stand (Appendix figure 12)**

- Zeg hardop: "Start Oefening 2" 1 keer aan het begin
- Ga staan vanuit de stoel en ga weer zitten (zonder je handen te gebruiken als dat lukt).
- Herhaal dit **10 keer**.



Appendix figure 11: Leg Extensions, Oefening 1

Appendix figure 12: Sit-to-Stand, Oefeing 2

- Rust: Wacht 1 minuut. Zeg hardop: "1 minuut rust” (ga naar de volgende pagina)

4. **Oefening 3: Lopen**

- Zeg hardop: "Start Oefening 3" 1 keer aan het begin
- Loop **5 rondjes** om de eettafel (of van de ene kant van de kamer naar de andere kant).
- Rust: Wacht 1 minuut. Zeg hardop: "1 minuut rust”

5. **Oefening 4: Traplopen**

- Zeg hardop: "Start Oefening 4" 1 keer aan het begin
- Loop de trap op en af.
- Herhaal dit **1 keer**.
- Rust: Wacht 1 minuut. Zeg hardop: "1 minuut rust”

Deel 3: Afronden en uitzetten

Stap 1 en 3 zijn alleen nodig als de powerbank is gebruikt tijdens de testen.

1. **Schakelaar UIT:** Zet de schakelaar achter de USB-C-poort weer om (van de blauwe stip *af*).
2. **Controle:** Check of het oranje/gele LED-lampje op de wearable nu volledig uit is.
3. **Opbergen:** Doe de wearable af en leg alles bij elkaar.

12.2 Python code comparing all exercises of non-diagnosed OA individual, and Healthy individual

```
import soundfile as sf
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.widgets import Slider, Button
import os
from scipy.signal import butter, filtfilt, resample, hilbert, find_peaks
import pandas as pd
from matplotlib.ticker import MultipleLocator

# ==========================================
# 1. CONFIGURATION
# ==========================================

# --- BESTAND 1 ---
FILE_1 = "2026-02-16_M4"
TITLE_1 = "Non-diagnosed OA individual"
CROP_START_1 = "1:30"
CROP_END_1   = "8:30"
EVENTS_1 = [
    ("Rest", "0:25"),
    ("Leg extensions", "0:31"),
    ("Rest", "0:58"),
    ("Sit-to-stand", "0:41"),
    ("Rest", "0:55"),
    ("Walking", "1:06"),
    ("Rest", "0:34"),
    ("Walking", "0:24"),
    ("Stairs", "0:45"),
]

# --- BESTAND 2 ---
FILE_2 = "2026-02-28_M7.wav" # <-- Vul hier je tweede bestand in
TITLE_2 = "Healty individual"
CROP_START_2 = "1:30"
CROP_END_2   = "8:30" # Kan je langer of korter maken, of None gebruiken
EVENTS_2 = [
    # Vul hier de events voor het tweede bestand in
    ("Rest", "0:30"),
    ("Leg extensions", "0:39"),
    ("Rest", "1:02"),
    ("Sit-to-stand", "0:39"),
    ("Rest", "1:05"),
    ("Walking", "0:48"),
    ("Rest", "0:51"),
    ("Stairs", "0:32"),
]

# --- ALGEMENE SETTINGS ---
PLOT_TITLE = "Comparison non-diagnosed OA individual vs healty indvidual, all exercises"
TARGET_FS = 48000
GLOBAL_OFFSET = 0.5
OUTPUT_FOLDER = "grafiek tweaken"

if not os.path.exists(OUTPUT_FOLDER):
    os.makedirs(OUTPUT_FOLDER)

# ==========================================
# 2. HULPFUNCTIE: TIJD OMREKENEN
# ==========================================
def parse_time_to_seconds(tijd_str):
    if tijd_str is None: return None
    try:
        tijd_str = str(tijd_str).replace(',', '.')
        if ':' in tijd_str:
            delen = tijd_str.split(':')
            minuten = int(delen[0])
            seconden = int(delen[1])
            return (minuten * 60) + seconden
        elif '.' in tijd_str:
            return float(tijd_str) * 60
        else:
            return int(tijd_str) * 60
    except:
        print(f"⚠ Kon tijd '{tijd_str}' niet lezen. Gebruik 0 seconden.")
        return 0

# ==========================================
# 3. SIGNAL PROCESSING FUNCTIONS
# ==========================================
```

```python
def butter_bandpass(lowcut, highcut, fs, order=6):
    nyq = 0.5 * fs
    low = max(0.01, lowcut / nyq)
    high = min(0.99, highcut / nyq)
    b, a = butter(order, [low, high], btype='band')
    return b, a

def get_hilbert_envelope(signal):
    return np.abs(hilbert(signal))

def load_audio_corrected(file_path, crop_start_str, crop_end_str):
    if not os.path.exists(file_path):
        print(f"⚠ Bestand niet gevonden: {file_path}")
        return None, None

    audio, fs = sf.read(file_path)
    if len(audio.shape) > 1: audio = audio[:, 0]

    if fs != TARGET_FS:
        audio = resample(audio, int(len(audio) * TARGET_FS / fs))
        fs = TARGET_FS

    # 1. Bepaal start
    t_start = parse_time_to_seconds(crop_start_str)
    t_start += GLOBAL_OFFSET
    start_sample = int(t_start * fs)

    # 2. Bepaal eind
    t_end = parse_time_to_seconds(crop_end_str)
    if t_end is not None and t_end > 0:
        end_sample = int(t_end * fs)
    else:
        end_sample = len(audio)

    # 3. Knippen maar
    start_sample = max(0, start_sample)
    end_sample = min(len(audio), end_sample)

    if start_sample >= len(audio):
        print(f"⚠ Starttijd ligt na het einde van het bestand: {file_path}")
        return None, None

    print(f"✂ Audio geknipt ({file_path}): {t_start:.1f}s tot {t_end if t_end
else 'Einde'}s")
    return audio[start_sample:end_sample], fs

# --- Hulpfunctie om uitschieters weg te halen ---
def remove_outliers(audio_data, threshold, width_sec, fs):
    if threshold >= 0.99: return audio_data

    cleaned = audio_data.copy()
    mask = np.abs(cleaned) > threshold
    if not np.any(mask): return cleaned

    width_samples = int(width_sec * fs)
    if width_samples > 0:
        kernel = np.ones(width_samples)
        expanded_mask = np.convolve(mask.astype(int), kernel, mode='same') > 0
        cleaned[expanded_mask] = 0.0
    else:
        cleaned[mask] = 0.0

    return cleaned

# --- Hulpfunctie om events te tekenen ---
def draw_events_on_axis(ax, events_list, max_time_min):
    huidige_tijd_sec = 0
    for activiteit, duur_str in events_list:
        duur_sec = parse_time_to_seconds(duur_str)
        eind_tijd_sec = huidige_tijd_sec + duur_sec

        start_tijd_min = huidige_tijd_sec / 60.0
        eind_tijd_min = eind_tijd_sec / 60.0
        midden_tijd_min = (huidige_tijd_sec + (duur_sec / 2)) / 60.0

        if start_tijd_min > max_time_min:
            break

        ax.axvline(x=eind_tijd_min, color='gray', linestyle='--', linewidth=1.5,
alpha=0.8)

        ax.text(midden_tijd_min, 0.95, activiteit,
                transform=ax.get_xaxis_transform(),
                rotation='vertical',
                ha='center', va='top',
                fontsize=10, fontweight='bold',
                color='#333333',
                bbox=dict(facecolor='white', alpha=0.6, edgecolor='none', pad=2.0))

        huidige_tijd_sec = eind_tijd_sec

# ========================================
# 4. INTERACTIVE PLOTTING (COMPARISON)
# ========================================

def interactive_tweaker_compare():
    # --- Data laden ---
    audio1, fs1 = load_audio_corrected(FILE_1, CROP_START_1, CROP_END_1)
    audio2, fs2 = load_audio_corrected(FILE_2, CROP_START_2, CROP_END_2)

    if audio1 is None or audio2 is None:
        print("Kan niet verder: een of beide bestanden missen.")
        return

    # Tijd-assen maken IN MINUTEN (kunnen dus verschillend van lengte zijn)
    t1 = (np.arange(len(audio1)) / fs1) / 60.0
```

```
    t2 = (np.arange(len(audio2)) / fs2) / 60.0

    max_time_min_1 = len(audio1) / fs1 / 60.0
    max_time_min_2 = len(audio2) / fs2 / 60.0

    # --- Plot opzetten (2 Subplots) ---
    fig, axes = plt.subplots(2, 1, figsize=(14, 10), sharex=True)
    plt.subplots_adjust(bottom=0.35, hspace=0.25) # Extra ruimte voor sliders

    # Lijsten om grafiek-elementen in op te slaan
    line_envs = []
    line_threshs = []
    line_peaks_list = []
    legends = []

    # Voorbereiding voor beide grafieken
    for i, (t_array, title, events_list, max_time) in enumerate(zip(
        [t1, t2], [TITLE_1, TITLE_2], [EVENTS_1, EVENTS_2], [max_time_min_1,
max_time_min_2]
    )):
        ax = axes[i]

        # Initiele lijnen
        l_env, = ax.plot(t_array, np.zeros(len(t_array)), color='#1d3557', lw=1.0,
label='Hilbert Envelope')
        l_thresh = ax.axhline(0, color='red', linestyle='--', alpha=0.7,
label='Threshold')
        l_peaks, = ax.plot([], [], "o", color='red', markersize=2, label='Detected
Peaks')

        line_envs.append(l_env)
        line_threshs.append(l_thresh)
        line_peaks_list.append(l_peaks)

        # Teken events
        draw_events_on_axis(ax, events_list, max_time)
        ax.plot([], [], color='gray', linestyle='--', label='Activity Separator')

        ax.set_title(title, fontweight='bold', fontsize=12)
        ax.set_ylabel("Amplitude (a.u.)", fontsize=12)
        ax.grid(True, alpha=0.2)

        # TIJDLIJN AANPASSEN
        ax.xaxis.set_major_locator(MultipleLocator(0.5)) # Zet een streepje elke
0.5 minuten
        ax.tick_params(labelbottom=True)                # Forceer dat de getallen
(labels) op álle grafieken staan

        # Legenda
        leg = ax.legend(loc='upper right', fontsize='small', framealpha=0.9)
        legends.append(leg)

    axes[-1].set_xlabel("Time (minutes)", fontsize=14)
    fig.suptitle(PLOT_TITLE, fontweight='bold', fontsize=16)
```

```
    # --- Sliders configureren (Nu in 2 kolommen) ---
    ax_low = plt.axes([0.15, 0.22, 0.3, 0.03])
    ax_high = plt.axes([0.15, 0.18, 0.3, 0.03])
    ax_std = plt.axes([0.15, 0.14, 0.3, 0.03])

    ax_out_thresh = plt.axes([0.55, 0.22, 0.3, 0.03])
    ax_out_width = plt.axes([0.55, 0.18, 0.3, 0.03])

    ax_save = plt.axes([0.85, 0.05, 0.1, 0.04])

    s_low = Slider(ax_low, 'Lowcut (Hz)', 100, 10000, valinit=1500, valstep=100)
    s_high = Slider(ax_high, 'Highcut (Hz)', 2000, 22000, valinit=20000,
valstep=500)
    s_std = Slider(ax_std, 'Threshold (STD)', 1.0, 10.0, valinit=10.0, valstep=0.5)

    s_out_thresh = Slider(ax_out_thresh, 'Outlier Thresh', 0.01, 1.0, valinit=0.30,
valstep=0.01)
    s_out_width = Slider(ax_out_width, 'Remove Width (s)', 0.0, 0.5, valinit=0.1,
valstep=0.01)

    b_save = Button(ax_save, 'Opslaan')

    # --- Update functie ---
    def update(val):
        low, high, std_m = s_low.val, s_high.val, s_std.val
        out_thresh, out_width = s_out_thresh.val, s_out_width.val

        if low >= high: return

        b, a = butter_bandpass(low, high, TARGET_FS)

        # --- 1. OUTLIERS VERWIJDEREN ---
        clean_audio1 = remove_outliers(audio1, out_thresh, out_width, TARGET_FS)
        clean_audio2 = remove_outliers(audio2, out_thresh, out_width, TARGET_FS)

        # --- 2. BEREKEN DE MASTER THRESHOLD OP BASIS VAN FILE 1 ---
        filt1 = filtfilt(b, a, clean_audio1)
        env1 = get_hilbert_envelope(filt1)
        master_threshold = np.mean(env1) + std_m * np.std(env1)

        # Verwerk beide opgeschoonde audio bestanden
        audio_data_clean = [clean_audio1, clean_audio2]
        t_arrays = [t1, t2]

        global_max_y = 0

        # --- 3. PAS DE THRESHOLD TOE OP BEIDE BESTANDEN ---
        for i in range(2):
            if i == 0:
                env = env1 # File 1 hebben we zojuist al berekend
            else:
                filt = filtfilt(b, a, audio_data_clean[i])
                env = get_hilbert_envelope(filt)
```

```
            # Pieken zoeken met de MASTER threshold
            peaks, _ = find_peaks(env, height=master_threshold,
distance=int(0.01*TARGET_FS))

            # Update Grafiek Data
            line_envs[i].set_ydata(env)
            line_threshs[i].set_ydata([master_threshold, master_threshold])
            line_peaks_list[i].set_data(t_arrays[i][peaks], env[peaks])

            # Update Legenda Tekst
            legends[i].get_texts()[1].set_text(f'Threshold:
{master_threshold:.4f}')
            legends[i].get_texts()[2].set_text(f'Detected Peaks (n={len(peaks)})')

            # Zoek de maximale Y waarde om ze op dezelfde schaal te houden
            global_max_y = max(global_max_y, np.max(env))

        # Zet dezelfde Y-schaal voor beide grafieken
        if global_max_y > 0:
            for ax in axes:
                ax.set_ylim(0, global_max_y * 1.3)

        # Titel updaten
        fig.suptitle(f"{PLOT_TITLE} | Bandpass: {int(low)}-{int(high)} Hz |
Threshold: Mean + {std_m}*STD | Outlier: {out_thresh:.2f}%", fontsize=14)
        fig.canvas.draw_idle()

    # --- Opslaan functie ---
    def save_plot(event):
        low, high, std_m, out_thresh = int(s_low.val), int(s_high.val), s_std.val,
s_out_thresh.val
        clean_name = PLOT_TITLE.replace(" ", "_").replace(":", "")
        filename = f"{clean_name}_{low}-
{high}Hz_plus{std_m}STD_Outlier_{out_thresh:.2f}.png"
        path = os.path.join(OUTPUT_FOLDER, filename)
        plt.savefig(path, dpi=300)
        print(f"☑ Grafiek opgeslagen: {path}")

    # Listeners koppelen
    s_low.on_changed(update)
    s_high.on_changed(update)
    s_std.on_changed(update)
    s_out_thresh.on_changed(update)
    s_out_width.on_changed(update)
    b_save.on_clicked(save_plot)

    # Eerste keer runnen
    update(None)
    plt.show()
if __name__ == "__main__":
    interactive_tweaker_compare()
```

12.3 Python code comparing sit-to-stand of a non-diagnosed OA individual, and a healthy individual with and without trousers

```
import soundfile as sf
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.widgets import Slider, Button
import os
from scipy.signal import butter, filtfilt, resample, hilbert, find_peaks
from matplotlib.ticker import MultipleLocator

# ========================================
# 1. CONFIGURATION (Vul hier je 4 bestanden in)
# ========================================

# --- BESTAND 1 (Linksboven - Bepaalt de Master Threshold) ---
FILE_1 = "2026-02-16_M4.wav"
TITLE_1 = "1. Without trousers non-diagnosed OA individual"
CROP_START_1 = "3:30"
CROP_END_1   = "3:50"
EVENTS_1 = [
    ("Sit-to-stand", "0:20"),
]

# --- BESTAND 2 (Rechtsboven) ---
FILE_2 = "2026-02-28_M7.wav"
TITLE_2 = "2. Without trousers healty individual"
CROP_START_2 = "3:50"
CROP_END_2   = "4:10"
EVENTS_2 = [
    ("Sit-to-stand", "0:20"),
]

# --- BESTAND 3 (Linksonder) ---
FILE_3 = "2026-02-16_M3r.wav"
TITLE_3 = "3. With trousers non-diagnosed OA individual"
CROP_START_3 = "1:08"
CROP_END_3   = "1:28"
EVENTS_3 = [
    ("Sit-to-stand", "0:20"),
]

# --- BESTAND 4 (Rechtsonder) ---
FILE_4 = "2026-02-28_M9.wav"
TITLE_4 = "4. With trousers healty individual"
CROP_START_4 = "0:33"
CROP_END_4   = "0:53"
EVENTS_4 = [
    ("Sit-to-stand", "0:20"),
```

```
]

# --- ALGEMENE SETTINGS ---
PLOT_TITLE = "Wearable Analysis: 4-Way Comparison"
TARGET_FS = 48000
GLOBAL_OFFSET = 0.0
OUTPUT_FOLDER = "grafiek tweaken 2x2"

if not os.path.exists(OUTPUT_FOLDER):
    os.makedirs(OUTPUT_FOLDER)

# ========================================
# 2. HULPFUNCTIES
# ========================================
def parse_time_to_seconds(tijd_str):
    if tijd_str is None: return None
    try:
        tijd_str = str(tijd_str).replace(',', '.')
        if ':' in tijd_str:
            delen = tijd_str.split(':')
            minuten = int(delen[0])
            seconden = int(delen[1])
            return (minuten * 60) + seconden
        elif '.' in tijd_str:
            return float(tijd_str) * 60
        else:
            return int(tijd_str) * 60
    except:
        print(f"⚠ Kon tijd '{tijd_str}' niet lezen. Gebruik 0 seconden.")
        return 0

def butter_bandpass(lowcut, highcut, fs, order=6):
    nyq = 0.5 * fs
    low = max(0.01, lowcut / nyq)
    high = min(0.99, highcut / nyq)
    b, a = butter(order, [low, high], btype='band')
    return b, a

def get_hilbert_envelope(signal):
    return np.abs(hilbert(signal))

def load_audio_corrected(file_path, crop_start_str, crop_end_str):
    if not os.path.exists(file_path):
        print(f"⚠ Bestand niet gevonden: {file_path}")
        return None, None

    audio, fs = sf.read(file_path)
    if len(audio.shape) > 1: audio = audio[:, 0]

    if fs != TARGET_FS:
        audio = resample(audio, int(len(audio) * TARGET_FS / fs))
        fs = TARGET_FS

    t_start = parse_time_to_seconds(crop_start_str)
    t_start += GLOBAL_OFFSET
    start_sample = int(t_start * fs)

    t_end = parse_time_to_seconds(crop_end_str)
    if t_end is not None and t_end > 0:
        end_sample = int(t_end * fs)
    else:
        end_sample = len(audio)

    start_sample = max(0, start_sample)
    end_sample = min(len(audio), end_sample)

    if start_sample >= len(audio):
        print(f"⚠ Starttijd ligt na het einde van het bestand: {file_path}")
        return None, None

    print(f"✂ Audio geknipt ({file_path}): {t_start:.1f}s tot {t_end if t_end
else 'Einde'}s")
    return audio[start_sample:end_sample], fs

def remove_outliers(audio_data, threshold, width_sec, fs):
    if threshold >= 0.99: return audio_data
    cleaned = audio_data.copy()
    mask = np.abs(cleaned) > threshold
    if not np.any(mask): return cleaned

    width_samples = int(width_sec * fs)
    if width_samples > 0:
        kernel = np.ones(width_samples)
        expanded_mask = np.convolve(mask.astype(int), kernel, mode='same') > 0
        cleaned[expanded_mask] = 0.0
    else:
        cleaned[mask] = 0.0
    return cleaned

def draw_events_on_axis(ax, events_list, max_time_sec):
    huidige_tijd_sec = 0
    for activiteit, duur_str in events_list:
        duur_sec = parse_time_to_seconds(duur_str)
        eind_tijd_sec = huidige_tijd_sec + duur_sec
        start_tijd_sec = huidige_tijd_sec
        midden_tijd_sec = huidige_tijd_sec + (duur_sec / 2)

        if start_tijd_sec > max_time_sec:
            break

        ax.axvline(x=eind_tijd_sec, color='gray', linestyle='--', linewidth=1.5,
alpha=0.8)
        ax.text(midden_tijd_sec, 0.95, activiteit,
                transform=ax.get_xaxis_transform(), rotation='vertical',
                ha='center', va='top', fontsize=10, fontweight='bold',
                color='#333333', bbox=dict(facecolor='white', alpha=0.6,
edgecolor='none', pad=2.0))
```

```
        huidige_tijd_sec = eind_tijd_sec

# ========================================
# 3. INTERACTIVE PLOTTING (2x2 GRID)
# ========================================

def interactive_tweaker_compare():
    # 1. Alle 4 de bestanden laden
    a1, fs1 = load_audio_corrected(FILE_1, CROP_START_1, CROP_END_1)
    a2, fs2 = load_audio_corrected(FILE_2, CROP_START_2, CROP_END_2)
    a3, fs3 = load_audio_corrected(FILE_3, CROP_START_3, CROP_END_3)
    a4, fs4 = load_audio_corrected(FILE_4, CROP_START_4, CROP_END_4)

    audios = [a1, a2, a3, a4]

    if any(a is None for a in audios):
        print("\n❌ Kan niet verder: Vul eerst alle 4 de juiste bestandsnamen in
bij de configuratie!")
        return

    # Tijd-assen berekenen
    fss = [fs1, fs2, fs3, fs4]
    t_arrays = [np.arange(len(a)) / f for a, f in zip(audios, fss)]
    max_times = [len(a) / f for a, f in zip(audios, fss)]

    titles = [TITLE_1, TITLE_2, TITLE_3, TITLE_4]
    events_lists = [EVENTS_1, EVENTS_2, EVENTS_3, EVENTS_4]

    # Plot opzetten (2x2)
    fig, axes = plt.subplots(2, 2, figsize=(16, 10), sharex=True)
    axes = axes.flatten()
    plt.subplots_adjust(bottom=0.35, hspace=0.3, wspace=0.15)

    line_envs, line_threshs, line_peaks_list, legends = [], [], [], []

    for i in range(4):
        ax = axes[i]

        l_env, = ax.plot(t_arrays[i], np.zeros(len(t_arrays[i])), color='#1d3557',
lw=1.0, label='Hilbert Envelope')
        l_thresh = ax.axhline(0, color='red', linestyle='--', alpha=0.7,
label='Threshold')
        l_peaks, = ax.plot([], [], "o", color='red', markersize=2, label='Detected
Peaks')

        line_envs.append(l_env)
        line_threshs.append(l_thresh)
        line_peaks_list.append(l_peaks)

        draw_events_on_axis(ax, events_lists[i], max_times[i])
        ax.plot([], [], color='gray', linestyle='--', label='Activity Separator')

        ax.set_title(titles[i], fontweight='bold', fontsize=12)
```

```
        # Assen benaming (alleen links en onder voor een clean look)
        if i % 2 == 0: ax.set_ylabel("Amplitude (a.u.)", fontsize=12)
        if i >= 2:     ax.set_xlabel("Time (seconds)", fontsize=12)

        ax.grid(True, alpha=0.2)

        # X-as formattering (om de 5 seconden)
        ax.xaxis.set_major_locator(MultipleLocator(5))
        ax.tick_params(labelbottom=True)

        leg = ax.legend(loc='upper right', fontsize='small', framealpha=0.9)
        legends.append(leg)

    # --- SLIDERS (2 kolommen) ---
    ax_low = plt.axes([0.15, 0.22, 0.3, 0.03])
    ax_high = plt.axes([0.15, 0.18, 0.3, 0.03])
    ax_std = plt.axes([0.15, 0.14, 0.3, 0.03])

    ax_out_thresh = plt.axes([0.55, 0.22, 0.3, 0.03])
    ax_out_width = plt.axes([0.55, 0.18, 0.3, 0.03])

    ax_save = plt.axes([0.85, 0.05, 0.1, 0.04])

    s_low = Slider(ax_low, 'Lowcut (Hz)', 100, 10000, valinit=1500, valstep=100)
    s_high = Slider(ax_high, 'Highcut (Hz)', 2000, 22000, valinit=20000,
valstep=500)
    s_std = Slider(ax_std, 'Threshold (STD)', 1.0, 10.0, valinit=10.0, valstep=0.5)

    s_out_thresh = Slider(ax_out_thresh, 'Outlier Thresh', 0.01, 1.0, valinit=0.30,
valstep=0.01)
    s_out_width = Slider(ax_out_width, 'Remove Width (s)', 0.0, 0.5, valinit=0.1,
valstep=0.01)

    b_save = Button(ax_save, 'Opslaan')

    # --- Update functie ---
    def update(val):
        low, high, std_m = s_low.val, s_high.val, s_std.val
        out_thresh, out_width = s_out_thresh.val, s_out_width.val
        if low >= high: return

        b, a = butter_bandpass(low, high, TARGET_FS)

        # 1. OUTLIERS VERWIJDEREN VOOR ALLE 4 DE BESTANDEN
        clean_audios = [remove_outliers(audio, out_thresh, out_width, TARGET_FS)
for audio in audios]

        # 2. BEREKEN DE MASTER THRESHOLD OP BASIS VAN FILE 1
        filt1 = filtfilt(b, a, clean_audios[0])
        env1 = get_hilbert_envelope(filt1)
        master_threshold = np.mean(env1) + std_m * np.std(env1)

        global_max_y = 0
```

```
        # 3. PAS TOE OP ALLE 4 DE BESTANDEN
        for i in range(4):
            if i == 0:
                env = env1
            else:
                filt = filtfilt(b, a, clean_audios[i])
                env = get_hilbert_envelope(filt)

            peaks, _ = find_peaks(env, height=master_threshold,
distance=int(0.01*TARGET_FS))

            line_envs[i].set_ydata(env)
            line_threshs[i].set_ydata([master_threshold, master_threshold])
            line_peaks_list[i].set_data(t_arrays[i][peaks], env[peaks])

            legends[i].get_texts()[1].set_text(f'Threshold):
{master_threshold:.4f}')
            legends[i].get_texts()[2].set_text(f'Detected Peaks: {len(peaks)}')

            global_max_y = max(global_max_y, np.max(env))

        # Zet overal de Y-as gelijk
        if global_max_y > 0:
            for ax in axes:
                ax.set_ylim(0, global_max_y * 1.3)

        fig.suptitle(f"{PLOT_TITLE} | BP: {int(low)}-{int(high)} Hz | Thresh: Mean
+ {std_m}*STD | Outlier Lim: {out_thresh:.2f}", fontsize=14)
        fig.canvas.draw_idle()

    # --- Opslaan functie ---
    def save_plot(event):
        low, high, std_m, out_thresh = int(s_low.val), int(s_high.val), s_std.val,
s_out_thresh.val
        clean_name = PLOT_TITLE.replace(" ", "_").replace(":", "")
        filename = f"{clean_name}_{low}-
{high}Hz_plus{std_m}STD_Outlier{out_thresh:.2f}.png"
        path = os.path.join(OUTPUT_FOLDER, filename)
        plt.savefig(path, dpi=300)
        print(f"☑ Grafiek opgeslagen: {path}")

    s_low.on_changed(update)
    s_high.on_changed(update)
    s_std.on_changed(update)
    s_out_thresh.on_changed(update)
    s_out_width.on_changed(update)
    b_save.on_clicked(save_plot)

    update(None)
    plt.show()

if __name__ == "__main__":
    interactive_tweaker_compare()
```

13 Validation temperature sensors Works-like prototype

13.1 Code graphs

```python
import pandas as pd
import matplotlib.pyplot as plt
import os

# ==========================================
# 1. CONFIGURATION (Pas dit hier aan)
# ==========================================
BESTANDSNAAM = "2026-02-16_M4.csv"

# --- HIER VUL JE DE ACTIVITEITEN IN ---
# EVENTS = [
#     ("Sitting", "7:10"),
#     ("Walking Inside", "0:50"),
#     ("Walking Outside", "2:06"),
#     ("Walking Inside", "0:50"),
#     ("Sitting", "7:00")
# ]
EVENTS = [
    ("Sitting", "5:35"),
    ("Sport", "7:30"),
    ("Sitting", "7:00")

]

# EVENTS = [
#     ("Sitting", "5:00"),
#     ("Walking", "8:00"),
#     ("Sitting", "5:00")

# ]


# Titel & Bereik
GRAFIEK_TITEL = "Temperature Analysis Test 3 without trousers"
Y_MIN = 10
Y_MAX = 35
SMOOTHING_WINDOW = 60

# Sensoren
SENSOR_CONFIG = {
    'T1': {'label': 'Environment sensor', 'color': "#B33939"},
    'T2': {'label': 'South',              'color': '#3498DB'},
    'T3': {'label': 'North Lateral',      'color': '#2E86C1'},
    'T4': {'label': 'North Medial',       'color': '#154360'}
}

# ==========================================
# 2. HULPFUNCTIE: TIJD OMREKENEN
# ==========================================
def parse_time_to_seconds(tijd_str):
    try:
        tijd_str = str(tijd_str).replace(',', '.')
        if ':' in tijd_str:
            delen = tijd_str.split(':')
        elif '.' in tijd_str:
            delen = tijd_str.split('.')
        else:
            return int(tijd_str) * 60

        minuten = int(delen[0])
        seconden = int(delen[1])
        return (minuten * 60) + seconden
    except:
        print(f"⚠ Kon tijd '{tijd_str}' niet lezen. Gebruik 0 seconden.")
        return 0

# ==========================================
# 3. DATA LADEN
# ==========================================
if not os.path.exists(BESTANDSNAAM):
    print(f"❌ Error: File not found: {BESTANDSNAAM}")
    exit()

print(f"📂 Loading file...")
df = pd.read_csv(BESTANDSNAAM)

if 'Tijd_Geschat_sec' in df.columns:
    df.rename(columns={'Tijd_Geschat_sec': 'Tijd_sec'}, inplace=True)
df['Tijd_sec'] = pd.to_numeric(df['Tijd_sec'], errors='coerce')

# --- NIEUW: TIJD OMREKENEN NAAR MINUTEN ---
df['Tijd_min'] = df['Tijd_sec'] / 60.0

beschikbare_sensoren = [s for s in SENSOR_CONFIG.keys() if s in df.columns]
if not beschikbare_sensoren:
    print("❌ No sensors found.")
    exit()

# ==========================================
# 4. GRAFIEK MAKEN
# ==========================================
fig, ax = plt.subplots(figsize=(14, 7))

# Lijnen plotten
for sensor_code in beschikbare_sensoren:
    config = SENSOR_CONFIG[sensor_code]
```

```
    smooth_data = df[sensor_code].rolling(window=SMOOTHING_WINDOW,
min_periods=1).mean()

    # AANGEPAST: We plotten nu 'Tijd_min' op de X-as
    ax.plot(df['Tijd_min'], smooth_data, color=config['color'], linewidth=2,
label=config['label'])

# ========================================
# 5. EVENTS TEKENEN (AANGEPAST NAAR MINUTEN)
# ========================================
huidige_tijd_sec = 0

print("\n⏱️ Events verwerken:")
for activiteit, duur_str in EVENTS:
    duur_sec = parse_time_to_seconds(duur_str)
    eind_tijd_sec = huidige_tijd_sec + duur_sec

    # Omrekenen naar minuten voor de grafiek
    eind_tijd_min = eind_tijd_sec / 60.0
    midden_tijd_min = (huidige_tijd_sec + (duur_sec / 2)) / 60.0

    # 1. Teken verticale lijn (in minuten)
    ax.axvline(x=eind_tijd_min, color='gray', linestyle='--', linewidth=1,
alpha=0.7)

    # 2. Zet de tekst (in minuten)
    ax.text(midden_tijd_min, Y_MAX - 1, activiteit,
            ha='center', va='top', fontsize=10, fontweight='bold',
            color='#555555', bbox=dict(facecolor='white', alpha=0.8,
edgecolor='none'))

    print(f"   -> {activiteit}: tot minuut {eind_tijd_min:.2f}")

    # Update de starttijd
    huidige_tijd_sec = eind_tijd_sec

# ========================================
# 6. OPMAAK
# ========================================
ax.set_title(GRAFIEK_TITEL, fontsize=16, fontweight='bold')

# AANGEPAST: Label van de as
ax.set_xlabel("Time (minutes)", fontsize=12)

ax.set_ylabel("Temperature (°C)", fontsize=12)
ax.set_ylim(Y_MIN, Y_MAX)
ax.grid(True, which='both', linestyle='--', linewidth=0.5, alpha=0.7)
ax.legend(loc='lower left', frameon=True)

plt.tight_layout()
plt.savefig(GRAFIEK_TITEL)
plt.show()
```

14 Validation Feels-like prototype

14.1 Interview guide

This test was done with 5 healthy Dutch test participants; therefore, the test questions are also in Dutch. These questions were put in a Form.

Vragen

1. Wat viel als eerst op toen je hem aandeed?
2. Hoe zwaar ervaar je de wearable tijdens het lopen? 1 niet voelbaar, 10 erg zwaar/belemmerend.
3. Beperkt de wearable je bewegingen (bijv. diep buigen, traplopen)? 1 volledige vrijheid, 10 sterk belemmerd.
4. Hoeveel minuten duurde het voordat je de wearable niet meer voelde zitten (tactile adaptation)?
5. Was het comfort aan het einde van de draagtijd (4 uur) hetzelfde als aan het begin?
6. In hoeverre verschoof of zakte de wearable af tijdens de test? 1 bleef perfect zitten, 10 zakte constant af
7. Hoeveel cm is hij verzakt?
8. Heb je tijdens het lopen de neiging gehad om de brace omhoog te trekken of recht te zetten?
9. Scoor de mate van drukpunten/knellingen. 1 geen drukpunten, 10 pijnlijke drukpunten.
10. Waar waren deze drukpunten, en hoeveel?
11. Zijn er zichtbare bulten of lijnen door je broek heen zichtbaar tijdens het dragen?
12. Scoor de temperatuur onder de wearable. 1 lekker koel, 10 erg warm broeierig.
13. Scoor de Zijn er na afloop rode plekken of irritaties op de huid zichtbaar?
14. Hoe intuïtief is het aantrekken (weet je direct wat boven/onder is)? 1 direct duidelijk, 10 grote puzzel
15. Hoeveel moeite kost het aantrekken fysiek (kracht/handigheid)? 1 heel makkelijk, 10 heel zwaar
16. Hoeveel seconden duurde het aantrekken ongeveer?
17. Was je bang om het materiaal te beschadigen (bijv. bij de gaten) tijdens het trekken? Wat zou dit gevoel verminderen?
18. Welke uitstraling vind je dat de wearable heeft?
19. Welke uitstraling vind je dat de wearable heeft? 1 zeer comfortabel, 10 ruw/irriterend

14.2 Results of comfort tests

1. Antropometrische Gegevens

- P1: Vrouw | Bovenbeen: 41 cm | Kuit: 36 cm
- P2: Vrouw | Bovenbeen: 44 cm | Kuit: 37 cm
- P3: Man | Bovenbeen: 44 cm | Kuit: 37,5 cm
- P4: Man | Bovenbeen: 44 cm | Kuit: 35,7 cm
- P5: Vrouw | Bovenbeen: 42 cm | Kuit: 40 cm

2. Eerste indruk & algemeen draagcomfort

- P1: Het zat lekkerder dan dat het eruitzag. Op het eerste moment geen vervelende punten; even wennen, maar geen specifieke pijnpunten.
- P2: Aan het begin zat het lekker en had ik nergens last van.
- P3: "Nee, hij was niet noticeable." Het werd comfortabeler naarmate ik hem langer droeg en eraan wende.
- P4: Niet per se heel oncomfortabel. Het onderste elastiek zat strakker dan de bovenste, en aan de zijkant zat het juist een beetje los.
- P5: Alleen de onderband knelde een beetje op mijn kuit. Verder voelde hij heel licht, onopvallend, niet afgekneld en niet te strak.

3. Bewustzijn

- P1: Toen ik afleiding had, voelde ik hem niet zo erg meer. Je voelt hem wel continu een beetje omdat je het niet gewend bent.
- P2: Ik vergat hem best snel; ik ging direct werken en had het toen niet meer door.
- P3: Zodra ik iets anders ging doen, dacht ik er niet meer aan. Duurde ongeveer 11 minuten.
- P4: Na zo'n 10 minuten, maar ik weet het niet meer precies.
- P5: Ja, vergeten dat ik hem aanhad. Na ongeveer een half uurtje had ik het niet meer door.

4. Comfort tijdens bewegen & drukpunten

- P1: Je voelt hem meebewegen, maar je hebt er geen last van. Alleen de naad van het bovenste elastiek (vooral achterop) was het ergste drukpunt.
- P2: Bij bewegen voel je wel even dat er iets op je knie zit, maar het is niet storend. Wel voelde ik de naad van het elastiek bovenop/achterop.
- P3: Geen last van bij het kruisen van rechts over links. Links over rechts merkte ik wel op, vanwege de hardware casing. Drukpuntje aan de binnenkant bij het elastiek.
- P4: Je voelt hem bijna niet, maar het bovenste elastiek blijft niet zo goed zitten. Geen echte drukpunten, behalve het blokje van de elektronica.
- P5: Soms knalde het harde deel (de casing) tegen de tafel, dat voel je dan wel. Nu ik erop let, voel ik de bovenband, de elastieken en het verdikte deel.

5. Stabiliteit

- P1: Hij bleef voor mijn gevoel op z'n plek zitten (uiteindelijke meting: 4 cm omhoog, 0,5 cm naar beneden).

- P2: Ik had de eerste keer bij het bewegen het idee dat hij afzakte, daarna niet meer (uiteindelijke meting: 2 cm omhoog, 0,7 cm naar beneden).
- P3: Ik wist niet dat hij naar beneden was gezakt totdat we gingen kijken (uiteindelijke meting: 12 cm naar beneden).
- P4: Ja, ik voelde dat hij afgleed. Hij lijkt te los aan de bovenkant. De antislip aan de onderkant werkt beter en is niet verschoven.
- P5: Bovenkant verschoof niet, onderkant verschoof ongeveer 1 cm. Had niet het gevoel dat hij bewoog tijdens het zitten.

6. Huidirritatie

- P1: Rode plekken bij het elastiek onder en het elastiek boven.
- P2: Bij het elastiek onder en boven wat rode plekken, maar geen echte irritatie.
- P3: Rode plekken bij het elastiek onder en boven.
- P4: Rode plekken bij het elastiek onder.
- P5: Rode plekken bij het elastiek onder en boven.

7. Gebruik onder kleding & warmte

- P1: Ik had een losse broek aan, dan zie je het kastje natuurlijk minder. De open achterkant is fijn voor buigen.
- P2: Geen duidelijke lijnen onder de kleding. Ik had een dunne broek aan, bij een dikke broek heb je er misschien nog minder last van. Neutraal qua warmte.
- P3: Ik had een strakkere broek aan, maar zag de wearable bijna niet; viel niet specifiek op. On the inside, it chafed my pants for a bit. De wearable is heel luchtig.
- P4: (Niet specifiek benoemd, gaf aan dat het blokje merkbaar was).
- P5: Voelt erg ademend.

8. Aantrekgemak

- P1: Duurde ~30 seconden. Snel duidelijk: knieschijf door het rondje en de lussen. Was voorzichtig, maar materiaal is stevig genoeg.
- P2: Duurde 10-15 seconden. Goed vasthouden om de voet erdoorheen te halen. Trok eerst op de verkeerde plek omhoog (beter aan de elastieken trekken). Logisch door knieschijfgat en casing.
- P3: Duurde 3 minuten de eerste keer, volgende keer 30 sec. Voet in het verkeerde gat gestoken. Het is erg licht, was bang dat het makkelijk kapot zou gaan.
- P4: Had het al een keer gezien, daardoor duidelijk dat de casing boven moest. Het blijft een beetje zoeken, maar de cirkel helpt. Best stevig.
- P5: Duurde ~30 seconden. Casing gaf duidelijk de bovenkant aan. Voelt dun en "flimsy", Was even bang de voet door de verkeerde gaten te stoppen, maar ziet er betrouwbaar uit.

9. Esthetiek

- P1: Wel nice, lekker simpele kleuren. Het elastiek ziet er minder comfortabel uit.
- P2: Ziet er sportief uit. Lekkere dunne stof, zit strak op de huid.

- P3: Ziet er goed en cool uit. Zou me niet schamen om dit in de sportschool te dragen.
- P4: Ziet er goed uit, zwart, prima. De buitenkant (zijkant) flubbert wel een beetje.
- P5: Heeft iets weg van een onderbroek/kniebrace, maar je kunt er wel mee sporten. Ziet er niet helemaal zacht/comfortabel uit doordat de randen misschien niet zijn afgewerkt.

10. Overige feedback & aanbevelingen van gebruikers

- P1: Bovenkant glijdt eigenlijk af en moet steviger, onderkant voelde ik bijna niet. Kijk naar verstelbaarheid van de band en verschillende maten. Kan hem wel 16 uur aanhouden als het moet.
- P2: Het midden is heeft een hoog comfort, maar het elastiek een wat minder die is wat ruwer.
- P3: Nadenken over hoe je hem aandoet.
- P4: Fijn dat hij licht is en niet superstrak zit, maar daardoor zakt hij misschien af. Elektronica mooier wegwerken.
- P5: Op de naden mag het wat dikker voelen voor meer stevigheid.

15 Validation Looks-like visuals

15.1 Test set-up

Concept validation knee wearable (online)

Goal: Assess whether the 'Final Design Vision' is acceptable for daily use (16 hours) by the target group, and to identify any barriers they perceive.

Presentation: A PowerPoint with an introduction and renders

Introduction

'Thank you for your time. I am working on my graduation project on osteoarthritis.'

Developing a wearable device that measures inflammation and crepitus sounds. Based on the hypothesis, explain what my project is about.

I will reveal pictures of what it might look like. Important: rather be honest than nice. If you find something ugly or awkward, let me know.

First impressions

Are there things you are currently thinking about that worry you?

How do you feel about wearing a wearable device like this for 16 hours?

Requirements that were used in the development (for context)

- That your freedom of motion is maintained.
- That it is not too large and can therefore be worn under your trousers.
- That it hardly slips down.
- Use a light elastic fabric so it does not get warmer underneath.
- That it takes as little time as possible to put on and take off.
- How it looks - discretion.
- Comfort.

The reveal

Show the renders of the Final Design Vision. Do not discuss the features yet.

Question 1: 'What is your first thought when you see this?' (Write down the first words: Futuristic? Medical? Ugly? Comfortable?)

Question 2: 'Would you be embarrassed to wear this in public (e.g. in the supermarket with shorts or long trousers)?'

Question 3: 'Does this look like a medical device (such as a hospital brace) or more like consumer electronics (such as a Fitbit/sports watch)?'

Deep dive: comfort & use

Now you will discuss some specific requirements in depth:

The 16-hour scenario: 'The goal is for this device to turn on in the morning and only turn off when you go to sleep. Looking at this picture, would you be open to that? Why/why not?'

Clothing: ‘For an accurate measurement, the device must be worn on the skin. That means it must be worn under long trousers or with shorts. Would that be a problem for you? Would you be scared to see any bumps in your trousers around the knee?’

Attachment: ‘How do you think it will stay in place? Are you afraid it will slip down while walking?’

Aesthetics: ‘Show wearable with colours, which one would you rather like? Would you rather have another colour?’

Donning the wearable: ‘Are there any worries you might have when putting the wearable on?’

The UI / app

‘Suppose you had to indicate in an app when you feel pain. Would you do that, or would you forget?’

‘Would you like to see your own data (graphs of your knee), or would that make you anxious?’

Conclusion

If you could change one thing about this design, what would it be?’

Any other insights or points you'd like to make?

15.2 Results

Introductie en eerste gedachtes

- **Feedback:** "Lijkt mij wel prima, ziet eruit als een kniebrace."
- **Vraag:** "Moet je de achterkant niet ook meten? Bij een knieblessure leg je vaak je hand op de bovenkant."
- **Feedback:** Het oogt functioneel, geen ondersteuning nodig, puur voor geluid/meting.

Vraag 2: Zou u zich schamen om dit in het openbaar te dragen (bijv. supermarkt)?

- **Feedback**: Nee

Analyse: comfort en gebruik

Thema Het 16-uursscenario: *Vraag: Het doel is 's ochtends aanzetten en pas bij het slapen uitzetten. Staat u daarvoor open?*

- **Feedback:** "Het ligt eraan hoe lang het traject duurt. Een week of 3 maanden? Als er een positieve factor is – dat je klachten kunt voorkomen – dan wil ik dat wel. Pijn is een goede motivator."
- **Nuance:** "Ik heb zelf nu niet zo vaak pijn, dan vind ik het prima, maar de noodzaak is lager. Maar als je artrose hebt en chronische pijn, wil je graag zo'n voorspelling hebben om klachten te voorkomen. Ik zou hem voor onderzoek best maanden om willen hebben; dagelijks is geen probleem als het helpt waarschuwen."

Thema kleding & zichtbaarheid: *Vraag: Het apparaat moet op de huid gedragen worden (onder broek of bij korte broek). Is dat een probleem of bent u bang voor zichtbare bobbels?*

- **Feedback:** "Nee, niet erg. Ik ben niet zo heel ijdel. Bij diabetes heb je ook zo'n sensor op je arm zitten. Ik sta er sneller voor open als ik er geen last van heb (geen irritatie). De een is gevoeliger voor de huid dan de ander, maar ik zou het doen als ik het nodig heb."

Thema bevestiging & afzakken: *Vraag: Bent u bang dat het tijdens het lopen naar beneden glijdt?*

- Feedback:
- "Met een strakkere broek zal hij minder snel zakken."
- "Eens per 4 uur even rechttrekken is acceptabel (tijdens een wc-momentje)."
- Tip: "Bij het skiën had ik een band met klittenband, dat was fijn want die kon je strakker zetten."
- Zorg: "De plaatsing van de hard casing (elektronica) is een risico; door het gewicht kan hij daar gaan zakken. Misschien de casing lager plaatsen?"

Thema esthetiek: *Vraag: Wat vind je van deze kleur?*

- **Feedback:** "Zwart is eigenlijk wel goed. Huidskleurig is 'net niet', dat ziet er vaak onprettig uit. Een kleurtje is grappig maar niet nodig. Het ziet er functioneel uit en dat is het ook."

Thema aantrekken (usability): *Vraag: Heeft u zorgen over het omdoen?*

- Feedback: "Gaat hij makkelijk op zijn plek? Moet de voet er doorheen?"
- **Zorg:** "Mensen die wat ouder zijn en stijver (artrose op andere plekken) moeten het ook aan kunnen. De meeste mensen met artrose zijn slecht ter been. Makkelijk houden is belangrijk!"

- **Suggestie:** Misschien werken met klittenband (openvouwen) in plaats van een kous waar je in moet stappen. Vooral met de gaten in de stof kunnen tenen blijven haken.

Gebruikersinterface & data

Thema pijn registreren (App): *Vraag: Zou u in een app aangeven wanneer u pijn voelt, of vergeet u dat?*

- **Feedback:** "Ik denk dat je vooral de bijzondere dingen moet weten. Niet elk tochtje naar de keuken, maar wel de grotere activiteiten (hardlopen) en de pijnbeleving daarbij."
- **Suggestie:** Koppeling met een Fitbit/smartwatch voor activiteitsdata zou handig zijn.

Thema data inzien: *Vraag: Wilt u uw eigen data zien*

- **Feedback:** "Ja, ik wil het zien. Vooral de voorspellende waarde: 'Ik heb een activiteit gedaan die belastend was voor mijn gewricht (geregistreerd), dus de volgende keer moet ik dat niet doen of rustiger aan doen.'"
- **Doel:** bijvoorbeeld Een App' die helpt bepalen of het verstandig is om morgen te tennissen.

Overige punten & conclusie

- **Sluiting:** "Twee klittenbandjes om hem makkelijk aan te doen en te kunnen verstellen. Hij is nu vrij hoog, dat ziet er wel fijn uit, maar verstellen is belangrijk."
- **Formaat:** "Als het draagcomfort hetzelfde blijft, zou het fijner zijn als hij wat compacter is."
- **Algemeen oordeel:** "Hij ziet er comfortabel uit en sluit mooi aan."

16 Bill of materials

Referance	Raw Materials/specification	Part Name	Part Description	Costs in Euro	Unit	estimate needed quantity on prototype	unit	Total Costs
Anti slip elastic silicons	Elastic, silicons	Anti slip elastic up	Siliconen elastiek 15mm - zwart	2,25	per Meter	38	cm	0,855
Anti slip elastic rubber	Elastic, rubber	Anti slip elastic down	Antislip elastiek - 20mm - zwart	2,15	per Meter	31	cm	0,6665
Spandex like	80% Polyamice, 20% Elastane	Fabric	Body fabric	18	per Meter'	30	cm^2	1,636
Cotton	100% Cotton	Non-elastic fabric	Cotton	4,99	per Meter'	10	cm^2	0,05
Wolvilt	100% Wool	Casing	Wolvilt - 20x30 cm - zwart	3,9	20/30 cm	10	cm^2	0,65
Velcrow	Velcrow	Velcrow	Klittenband naaibaar 20 mm/ 1,2	5,25	1,25 Mete	1	cm^2	0,042
Conductive Yarn	Silvercoated, Vectran	Wire	LIBERATOR® 40 Highly Conductive	85	30 Meter	3	meter	8,5
NTC's		Temperature Sensors	RS PRO Thermistor, 10kΩ Resistan	4,73	1	5		23,65
Cheep MEMS		Acoustic Sensor Breakc	I2S MEMS Microphone Breakout -	7,2	1	1		7,2
Expensive MEMS		Acoustic Sensor flexibl	EV_ICS-43434-FX	55,9	1	1		55,9
Foam		Sponge foam for acoustic sensor		5,56	20 cm/ 10	1	cm^3	0,00556
Insulated copper wire Thin		Wire connection Acous	Thin Prototyping & Repair Wire - 2	7,5	200 Meter	50	cm	0,01875
Insulated copper wire Thi	PVC insulation	Wire connection XIAO	RS PRO Black 0.26 mm² Equipmen	85,52	100 Meter	10	cm	0,08552
Xiao ESP32 S3		Processor	Seeed Studio XIAO ESP32-S3	7,49	1	1		7,49
LiPo battery		Power	Lithium Ion Polymer Battery - 3.7	5,95	1	1		5,95
SD module		Storage module	MicroSD card breakout board+	7,5	1	1		7,5
Micro SD card		Storage card	SanDisk 16GB microSD Card	10	1	1		10
						Expensive MEMS		130,1993
						Cheep MEMS		74,29933

Referance	Link
Anti slip elastic silicons	https://www.fournituren4fun.eu/siliconen-elastiek-15mm-zwart
Anti-slip elastic rubber	https://www.fournituren4fun.eu/antislip-elastiek-20mm-zwart-p10315
Spandex like	t'stoffigpakhuis Delft
Cotton	https://www.delappenkraam.nl/stevig-katoen-donkerblauw.html#moreinfo
Wolvilt	https://pipoos.com/wolvilt-nr-40-zwart-88141
Velcrow	https://pipoos.com/klittenband-naaibaar-20-mm-1-25-meter-zwart-68369
Conductive Yarn	https://www.amazon.com/LIBERATOR-Highly-Conductive-Thread-Textiles/dp/B08X8MT2FH
NTC's	https://nl.rs-online.com/web/p/thermistor-ics/0151237?gb=s
Cheep MEMS	https://www.amazon.com/Ics-43434-Omnidirectional-Microphone-Precision-Detector/dp/B0FTMHNW9W
Expensive MEMS	https://eu.mouser.com/ProductDetail/TDK-InvenSense/EV_ICS-43434-FX?qs=u4fy%2FsgLU9PLLNAG7v5nUg%3D%3D&msclkid=4a795b6c03b8137ce92f324b0a6670db&utm_source=bing&utm_medium=cpc&utm_campaign=DSAs+Other+Suppliers+Netherlands+English&utm_term=%2FProductDetail%2F+In+Stock%3A&utm_content=Other+Suppliers+DSAs%3A+PDP
Foam	https://www.amazon.com/Carlisle-36550100-Commercial-Cleaning-Washing/dp/B07FGHF62T/ref=sr_1_3?dib=eyJ2IjoiMSJ9.nRyd1AHP4FOGL9CQaTmW7bDHsvj4ha2mTVjoiei140IAX90rKvcGqYxGen7EuJ961zpwDhpDmbIYGUjdbrQbiP3TTyMWlyJDAqt3hyukQYHMmlQOIviiFofGSW9KV6VXDEwBxCqHNPbgf_rFcNKwGVqbLDtTgz3KgwYOYQdpG4C5j22841nudkgt3oiy_a08Sk91C_1Nl3dt7gQ4wmS-Bk5Id-taxttPJ5M_f8_ztYebLjVjHWPdNdeToDIvR0qdMvwQyaxSZzT9ICFs64h1a43Im6FGcc-CLFWr8vvjpYo.sXCiOkk6UD0bQuU2g_aPDPl_LwaGWhurc1P9UO_UsKs&dib_tag=se&keywords=foam%2Bsponge&qid=1770024402&sr=8-3&th=1
Insulated copper wire Thin	https://www.adafruit.com/product/1446
Insulated copper wire Thick	https://ae.rsdelivers.com/product/rs-pro/rs-pro-black-026-mm-equipment-wire-23-awg-1/06-mm-100m-pvc-insulation/7125291
Xiao ESP32 S3	https://www.seeedstudio.com/XIAO-ESP32S3-p-5627.html
LiPo battery	https://www.adafruit.com/product/1317
SD module	https://www.adafruit.com/product/254
Micro SD card	https://www.amazon.com/SanDisk-COMINU024966-16GB-microSD-Card/dp/B004KSMXVM

17 Expert validation

Doel van de Validatie Het doel van dit semi-gestructureerde expert-interview was het valideren van de klinische relevantie, het ontwerp en de financiële haalbaarheid (viability) van het *proof-of-concept* voor toekomstig medisch onderzoek.

Aanbevelingen die kwamen uit eerdere testen zijn ook besproken als:

- Wrap-around design (klittenband). Dit is makkelijker aan te trekken voor ouderen.,
- Verplaatsing van de casing naar beneden.
- Het verzenden van de data zal een keer per dag gebeuren als de wearable zal worden opgeladen.
- Activiteitn kunnen worden geconnect aan een smart watch
- En belangrijke notities als pijnperiode zouden de patiënten via een app kunnen delen.

Testopzet & procedure

Het interview was opgebouwd uit drie evaluatiemomenten deze worden besproken met de expert

- **Design & usability (desirability):** Presentatie van het looks-like/feels-like prototype en de resultaten van eerdere gebruikerstests. Vraagstelling: Voldoet de vormgeving aan de verwachtingen voor de doelgroep?
- **Techniek & data (feasibility):** Presentatie van de hardware (casing), sensorplaatsing en de uitgeprinte datagrafieken (akoestisch en thermisch). Vraagstelling: Levert dit voldoende en accuraat bewijs voor klinische relevantie?
- **Kosten & haalbaarheid (viability):** Presentatie van de kostencalculatie (geschat op ~€132 voor de dure sensor, €76 voor de goedkope sensor). Vraagstelling: Is deze prijs realistisch en schaalbaar voor een klinische trial?

Resultaten en feedback van de expert

Thema 1: Wearable design & usability

- **Esthetiek en acceptatie:** De sportieve, zwarte uitstraling is positief.
- **Doelgroep (artrose):** Het huidige *slip-on-design* is inderdaad een obstakel. De doelgroep is vaak op leeftijd en heeft vaker overgewicht. Een *wrap-around* design (met klittenband) is beter en zo ook het aanbieden van verschillende maten.

Thema 2: Sensordaten und klinische tests

- **Waardering:** Het is een absoluut goed startpunt. Het 's nachts opladen van het apparaat is een zeer praktische en goede keuze.
- **Akoestiek (crepitaties):** Omgevingsgeluid van de broek blijft een uitdaging. De volgende technische stap vereist patroonherkenning om daadwerkelijke crepitaties te filteren van ruis.
- **Temperatuur:** moet verder worden uitgewerkt om echt te gebruiken. Wat betekenen bepaalde temperatuurverschillen?
- **Toekomstig testprotocol:** Om de data goed te valideren moeten specifieke handelingen getest worden, zoals traplopen en squatten. De ideale onderzoeksopzet is: eerst gezonde mensen testen, daarna patiënten met OA, gevolgd door een correlatie tussen de gemeten data en de subjectieve pijnrapportage van de patiënt over tijd. Er moet ook rekening gehouden worden met de invloed van spieraanspanning op de sensoren.

- Er moet echter wel nog een data-analyse naar kijken en validatietesten zijn nog een optie, om misschien de temperatuur en het geluid ook subjectief te gaan meten. En hieruit dan te bepalen of het het waard is om hier verder in te investeren.

Thema 3: Kosten en viability

- **Onderzoeksbudget:** Een stukprijs van € 250 is veel geld voor klinisch onderzoek. Een kleine testgroep (bijv. 30 apparaten voor 100 wisselende patiënten over 3 maanden) kost al snel € 7.500.
- **Cost-Benefit:** De levensvatbaarheid hangt volledig af van de *cost-benefit ratio*. De wearable verdient zijn waarde terug in bespaarde zorgkosten als hij in een vroege fase een 'trigger' (aankomende flare-up) kan voorspellen, zodat er tijdig kan worden ingegrepen. Met de voorgestelde technische stappen wordt het concept als zeer levensvatbaar gezien.

18 Arduino codes for battery recommendation

18.1 SD writing, Wi-Fi/No SD in SD module

```
#include <driver/i2s.h>
#include <WiFi.h>
#include <SD.h>
#include <SPI.h>
#include "FS.h"
#include "esp_heap_caps.h"

// ========================================
// CONFIGURATIE
// ========================================
const char* ssid     = "xxxx";
const char* password = "xxxx";
WiFiServer server(5001);

#define I2S_WS       5   // D4
#define I2S_BCK      6   // D5
#define I2S_DATA     43  // D6
#define SD_CS        44  // D7

const int ntcPins[] = {A0, A1, A2, A3};

#define SAMPLE_RATE    48000
#define BUFFER_SAMPLES 1024

// We gebruiken geen vaste naam meer, maar een variabele buffer
char currentFilename[32] = "/storage0.bin";

// ========================================
// DATA STRUCTUUR (PACKED = ESSENTIEEL)
// ========================================
struct __attribute__((packed)) DataPacket {
    uint16_t syncWord;        // NIEUW: Altijd 0xAAAA
    uint8_t packetType;
    uint32_t sequenceNumber;
    union {
        int16_t audio[BUFFER_SAMPLES];
        float temps[4];
    } payload;
};

QueueHandle_t dataQueue = NULL;
uint32_t globalSequenceCount = 0;

i2s_config_t i2s_config = {
    .mode = (i2s_mode_t)(I2S_MODE_MASTER | I2S_MODE_RX),
    .sample_rate = SAMPLE_RATE,
    .bits_per_sample = I2S_BITS_PER_SAMPLE_32BIT,
    .channel_format = I2S_CHANNEL_FMT_ONLY_LEFT,
    .communication_format = I2S_COMM_FORMAT_I2S,
    .intr_alloc_flags = ESP_INTR_FLAG_LEVEL1,
    .dma_buf_count = 8,
    .dma_buf_len = 512,
    .use_apll = false
};

// ========================================
// NTC FUNCTIES
// ========================================
float Read_Single_NTC(int pin) {
    float Vntc = (analogRead(pin) * 3.3) / 4095.0;
    if (Vntc < 0.05) return 0.0;
    float Rntc = 10000.0 * ((3.3 / Vntc) - 1.0);
    float a = 639.5, b = -0.1332, c = -162.5;
    return a * pow(Rntc, b) + c;
}

float Get_Avg_NTC(int pin) {
    float sum = 0;
    for(int i=0; i<10; i++) {
        sum += Read_Single_NTC(pin);
        delayMicroseconds(100);
    }
    return sum / 10.0;
}

// ========================================
// HULPFUNCTIE: BESTANDSNAAM KIEZEN
// ========================================
void determineFilename() {
    int counter = 0;
    while (true) {
        // Maak bestandsnaam: storage0.bin, storage1.bin, etc.
        snprintf(currentFilename, 32, "/storage%d.bin", counter);

        // Als dit bestand NIET bestaat, is dit onze nieuwe file!
        if (!SD.exists(currentFilename)) {
            Serial.print("🆕 Nieuw bestand aangemaakt: ");
            Serial.println(currentFilename);
            break;
        }
        // Als het wel bestaat, probeer de volgende
        counter++;

        // Veiligheid: stop bij 999 files
        if (counter > 999) break;
    }
}
```

```
// ========================================
// TAAK 1: AUDIO (CORE 1)
// ========================================
void audioTask(void *pvParameters) {
    static int32_t raw_buffer[BUFFER_SAMPLES];
    static DataPacket pkg;
    pkg.syncWord = 0xAAAA;
    while (true) {
        size_t bytes_read = 0;
        i2s_read(I2S_NUM_0, raw_buffer, sizeof(raw_buffer), &bytes_read,
portMAX_DELAY);

        if (bytes_read > 0) {
            pkg.packetType = 0;
            pkg.sequenceNumber = globalSequenceCount++;

            int samples_read = bytes_read / 4;
            for (int i = 0; i < samples_read; i++) {
                pkg.payload.audio[i] = (int16_t)(raw_buffer[i] >> 16);
            }
            xQueueSend(dataQueue, &pkg, 0);
        }
    }
}



// ========================================
// TAAK 2: MANAGER (EN SD, EN WIFI)
// ========================================
void dataManagerTask(void *pvParameters) {
    static DataPacket rx_pkg;
    static DataPacket temp_pkg;
    unsigned long lastTempMillis = 0;
    temp_pkg.syncWord = 0xAAAA;
    File dataFile;
    bool isFileOpen = false;
    int packetsWrittenSinceSave = 0; // Voor de veiligheids-flush

    while (true) {
        // --- A. TEMP METING ---
        if (millis() - lastTempMillis >= 1000) {
            lastTempMillis = millis();
            temp_pkg.packetType = 1;
            temp_pkg.sequenceNumber = globalSequenceCount;
            for(int i=0; i<4; i++) temp_pkg.payload.temps[i] =
Get_Avg_NTC(ntcPins[i]);
            xQueueSend(dataQueue, &temp_pkg, 0);
            Serial.printf("Temp Sync: %d\n", temp_pkg.sequenceNumber);
        }

        WiFiClient client = server.available();

        if (client) {
            // 1. Sluit bestand als het nog open stond van de offline modus
            if (isFileOpen) { dataFile.close(); isFileOpen = false; }

            Serial.println("👀 Client kijkt mee via WiFi");
            digitalWrite(LED_BUILTIN, HIGH);
            client.setNoDelay(true);

            // 2. Historie sturen (Wat al op de SD stond)
            // We sturen dit ALLEEN naar de client, we laten het op de SD staan!
            if (SD.exists(currentFilename)) {
                File readFiles = SD.open(currentFilename, FILE_READ);
                uint8_t temp_buf[sizeof(DataPacket)];
                while (readFiles.available() && client.connected()) {
                    readFiles.read(temp_buf, sizeof(DataPacket));
                    client.write(temp_buf, sizeof(DataPacket));

                    vTaskDelay(1);
                }
                readFiles.close();

            }

            // 3. Live Streamen (EN Opslaan)
            // We openen het bestand nu in APPEND mode om nieuwe data toe te voegen
            dataFile = SD.open(currentFilename, FILE_APPEND);
            if (dataFile) isFileOpen = true;
            packetsWrittenSinceSave = 0;

            while (client.connected()) {
                if (xQueueReceive(dataQueue, &rx_pkg, pdMS_TO_TICKS(10))) {

                    // STAP A: ALTIJD EERST OPSLAAN OP SD (Master Copy)
                    if (isFileOpen) {
                        dataFile.write((uint8_t *)&rx_pkg, sizeof(DataPacket));
                        packetsWrittenSinceSave++;

                        // Veiligheid: elke seconde saven (flush)
                        if (packetsWrittenSinceSave >= 50) {
                            dataFile.flush();
                            packetsWrittenSinceSave = 0;
                        }
                    }

                    // STAP B: DAARNA NAAR WIFI STUREN (Kijkgaatje)
                    // Als dit mislukt is dat niet erg, want het staat al op SD
(Stap A)
                    client.write((uint8_t *)&rx_pkg, sizeof(DataPacket));
                }
            }

            // Client is weggevallen
            client.stop();
            Serial.println("👌 Client verbroken, ga door met opnemen...");
```

```
        }
        else {
            // --- OFFLINE MODE ---


            if (WiFi.status() == WL_CONNECTED) digitalWrite(LED_BUILTIN,
(millis()/500)%2);
            else digitalWrite(LED_BUILTIN, (millis()/100)%2);

            if (xQueueReceive(dataQueue, &rx_pkg, pdMS_TO_TICKS(10))) {
                if (!isFileOpen) {
                    dataFile = SD.open(currentFilename, FILE_APPEND);
                    if (dataFile) isFileOpen = true;
                    packetsWrittenSinceSave = 0;
                }

                if (isFileOpen) {
                    dataFile.write((uint8_t *)&rx_pkg, sizeof(DataPacket));
                    packetsWrittenSinceSave++;

                    if (packetsWrittenSinceSave >= 50) {
                        dataFile.flush();
                        packetsWrittenSinceSave = 0;
                    }
                }
            } else {
                if (isFileOpen) {
                    dataFile.close();
                    isFileOpen = false;
                }
            }
        }
        vTaskDelay(1);
    }
}

// ========================================
// SETUP
// ========================================
unsigned long lastWifiRetry = 0;
const unsigned long wifiInterval = 60000; // 60.000 ms = 1 minuut

void setup() {
    Serial.begin(115200);
    analogReadResolution(12);
    pinMode(LED_BUILTIN, OUTPUT);
    delay(2000);

    Serial.println("\n--- START SYSTEEM (AUTO FILENAME) ---");

    WiFi.disconnect(true);
    delay(500);
    WiFi.mode(WIFI_STA);
    WiFi.begin(ssid, password);
```

```
    dataQueue = xQueueCreateWithCaps(200, sizeof(DataPacket), MALLOC_CAP_SPIRAM);
    if (dataQueue == NULL) { Serial.println("❌ FOUT: Geen PSRAM Queue!");
while(1); }

    SPI.begin(7, 8, 9, 44);
    if (!SD.begin(SD_CS)) {
        Serial.println("⚠️ Geen SD kaart!");
    } else {
        Serial.println("☑️ SD Kaart OK.");
        // HIER BEPALEN WE DE BESTANDSNAAM VOOR DEZE SESSIE
        determineFilename();
    }

    i2s_driver_install(I2S_NUM_0, &i2s_config, 0, NULL);
    i2s_pin_config_t pins = { .bck_io_num = (gpio_num_t)I2S_BCK, .ws_io_num =
(gpio_num_t)I2S_WS, .data_out_num = I2S_PIN_NO_CHANGE, .data_in_num =
(gpio_num_t)I2S_DATA };
    i2s_set_pin(I2S_NUM_0, &pins);

    server.begin();

    xTaskCreatePinnedToCore(audioTask, "Audio", 20000, NULL, 10, NULL, 1);
    xTaskCreatePinnedToCore(dataManagerTask, "Manager", 20000, NULL, 5, NULL, 0);

    Serial.println("🚀 Meting is gestart!");
    Serial.print("Schrijft naar: ");
    Serial.println(currentFilename);
}

void loop() {
    static bool connected = false;
    unsigned long currentMillis = millis();

    // 1. Check of we verbonden zijn
    if (WiFi.status() == WL_CONNECTED) {
        if (!connected) {
            connected = true;
            Serial.print("\n☑️ WiFi Verbonden! IP: ");
            Serial.println(WiFi.localIP());
            // Zet de LED constant AAN als er wifi is
            digitalWrite(LED_BUILTIN, HIGH);
        }
    }
    else {
        // 2. NIET verbonden. Is het tijd voor een nieuwe poging? (Elke minuut)
        if (connected) {
            connected = false;
            Serial.println("\n⚠️ WiFi Verbinding verloren.");
            digitalWrite(LED_BUILTIN, LOW);
        }

        if (currentMillis - lastWifiRetry >= wifiInterval) {
```

```
            lastWifiRetry = currentMillis;

            Serial.println("🔄 1 minuut voorbij: Opnieuw proberen te verbinden met
WiFi...");

            // Probeer opnieuw te verbinden (niet-blokkerend)
            WiFi.disconnect();
            WiFi.reconnect();
            // OF gebruik: WiFi.begin(ssid, password); als reconnect niet werkt
        }
    }

    // Kleine vertraging om de CPU in de loop taak rust te geven (audio draait toch
op andere core)
    vTaskDelay(100);
}
```

18.2 No SD programmed, Wi-Fi connection

```
#include <driver/i2s.h>
#include <WiFi.h>
#include "esp_heap_caps.h"

// ==========================================
// CONFIGURATIE
// ==========================================
const char* ssid     = "xxxx";
const char* password = "xxxx";
WiFiServer server(5001);

#define I2S_WS       5   // D4
#define I2S_BCK      6   // D5
#define I2S_DATA     43  // D6
// SD_CS is verwijderd want we gebruiken geen SD meer

const int ntcPins[] = {A0, A1, A2, A3};

#define SAMPLE_RATE    48000
#define BUFFER_SAMPLES 1024

// ==========================================
// DATA STRUCTUUR
// ==========================================
struct __attribute__((packed)) DataPacket {
    uint16_t syncWord;
    uint8_t packetType;
    uint32_t sequenceNumber;
    union {
        int16_t audio[BUFFER_SAMPLES];
        float temps[4];
    } payload;
};

QueueHandle_t dataQueue = NULL;
uint32_t globalSequenceCount = 0;

i2s_config_t i2s_config = {
    .mode = (i2s_mode_t)(I2S_MODE_MASTER | I2S_MODE_RX),
    .sample_rate = SAMPLE_RATE,
    .bits_per_sample = I2S_BITS_PER_SAMPLE_32BIT,
    .channel_format = I2S_CHANNEL_FMT_ONLY_LEFT,
    .communication_format = I2S_COMM_FORMAT_I2S,
    .intr_alloc_flags = ESP_INTR_FLAG_LEVEL1,
    .dma_buf_count = 8,
    .dma_buf_len = 512,
    .use_apll = false
};

// ==========================================
// NTC FUNCTIES
// ==========================================
float Read_Single_NTC(int pin) {
    float Vntc = (analogRead(pin) * 3.3) / 4095.0;
    if (Vntc < 0.05) return 0.0;
    float Rntc = 10000.0 * ((3.3 / Vntc) - 1.0);
    float a = 639.5, b = -0.1332, c = -162.5;
    return a * pow(Rntc, b) + c;
}

float Get_Avg_NTC(int pin) {
    float sum = 0;
    for(int i=0; i<10; i++) {
        sum += Read_Single_NTC(pin);
        delayMicroseconds(100);
    }
    return sum / 10.0;
}

// ==========================================
// TAAK 1: AUDIO (CORE 1)
// ==========================================
void audioTask(void *pvParameters) {
    static int32_t raw_buffer[BUFFER_SAMPLES];
    static DataPacket pkg;
    pkg.syncWord = 0xAAAA;

    while (true) {
        size_t bytes_read = 0;
        i2s_read(I2S_NUM_0, raw_buffer, sizeof(raw_buffer), &bytes_read,
portMAX_DELAY);

        if (bytes_read > 0) {
            pkg.packetType = 0; // 0 = Audio
            pkg.sequenceNumber = globalSequenceCount++;
```

```
            int samples_read = bytes_read / 4;
            for (int i = 0; i < samples_read; i++) {
                pkg.payload.audio[i] = (int16_t)(raw_buffer[i] >> 16);
            }
            xQueueSend(dataQueue, &pkg, 0);
        }
    }
}

// ========================================
// TAAK 2: MANAGER (CORE 0) - NU ZONDER SD
// ========================================
void dataManagerTask(void *pvParameters) {
    static DataPacket rx_pkg;
    static DataPacket temp_pkg;
    unsigned long lastTempMillis = 0;
    temp_pkg.syncWord = 0xAAAA;

    // Teller om serial monitor niet te overspoelen met audio meldingen
    int audioPrintCounter = 0;

    while (true) {
        // --- A. TEMP METING ---
        if (millis() - lastTempMillis >= 1000) {
            lastTempMillis = millis();
            temp_pkg.packetType = 1; // 1 = Temp
            temp_pkg.sequenceNumber = globalSequenceCount;
            for(int i=0; i<4; i++) temp_pkg.payload.temps[i] =
Get_Avg_NTC(ntcPins[i]);

            // Stuur naar queue zodat WiFi hem ook krijgt
            xQueueSend(dataQueue, &temp_pkg, 0);

            // PRINT NAAR SERIAL MONITOR
            Serial.printf("🌡️ TEMP: T1:%.1f | T2:%.1f | T3:%.1f | T4:%.1f\n",
                          temp_pkg.payload.temps[0], temp_pkg.payload.temps[1],
                          temp_pkg.payload.temps[2], temp_pkg.payload.temps[3]);
        }

        // --- B. WIFI CHECK ---
        WiFiClient client = server.available();

        // Als er data in de wachtrij staat (Audio of Temp)
        if (xQueueReceive(dataQueue, &rx_pkg, pdMS_TO_TICKS(10))) {

            // 1. Als er een WiFi client is, stuur data door
            if (client && client.connected()) {
                client.setNoDelay(true); // Snellere overdracht
                client.write((uint8_t *)&rx_pkg, sizeof(DataPacket));
            }

            // 2. Print status naar Serial Monitor
            if (rx_pkg.packetType == 0) { // Audio packet
                audioPrintCounter++;
                // Print elke 100 pakketjes even een stipje of tekst, anders crasht
serial monitor door snelheid
                if (audioPrintCounter >= 100) {
                    Serial.print("."); // Laat zien dat audio loopt
                    audioPrintCounter = 0;
                }
            }
        }
    }
}

// ========================================
// SETUP
// ========================================
void setup() {
    Serial.begin(115200);
    analogReadResolution(12);
    pinMode(LED_BUILTIN, OUTPUT);
    delay(2000);

    Serial.println("\n--- START SYSTEEM (GEEN SD - ALLEEN WIFI/SERIAL) ---");

    WiFi.disconnect(true);
    delay(500);
    WiFi.mode(WIFI_STA);
    WiFi.begin(ssid, password);

    // Wacht niet eindeloos op wifi, ga gewoon door
    Serial.print("Verbinden met WiFi");

    dataQueue = xQueueCreateWithCaps(200, sizeof(DataPacket), MALLOC_CAP_SPIRAM);
    if (dataQueue == NULL) { Serial.println("❌ FOUT: Geen PSRAM Queue!");
while(1); }

    // I2S Starten
    i2s_driver_install(I2S_NUM_0, &i2s_config, 0, NULL);
    i2s_pin_config_t pins = { .bck_io_num = (gpio_num_t)I2S_BCK, .ws_io_num =
(gpio_num_t)I2S_WS, .data_out_num = I2S_PIN_NO_CHANGE, .data_in_num =
(gpio_num_t)I2S_DATA };
    i2s_set_pin(I2S_NUM_0, &pins);

    server.begin();

    // Taken starten
    xTaskCreatePinnedToCore(audioTask, "Audio", 20000, NULL, 10, NULL, 1);
    xTaskCreatePinnedToCore(dataManagerTask, "Manager", 20000, NULL, 5, NULL, 0);

    Serial.println("\n🚀 Meting is gestart! Check Serial Monitor voor
temperaturen.");
}

void loop() {
    static bool connected = false;
```

```
    // WiFi status update in Serial
    if (WiFi.status() == WL_CONNECTED) {
        if (!connected) {
            connected = true;
            Serial.print("\n☑ WiFi Verbonden! IP: ");
            Serial.println(WiFi.localIP());
            digitalWrite(LED_BUILTIN, HIGH);
        }
    } else {
        if (connected) {
            connected = false;
            Serial.println("\n⚠ WiFi Verbinding verbroken!");
            digitalWrite(LED_BUILTIN, LOW);
        }
    }

    vTaskDelay(1000);
}
```

18.3 No SD programming, no WiFi connection

```
#include <driver/i2s.h>
#include "esp_heap_caps.h"

// ========================================
// CONFIGURATIE (PINNEN)
// ========================================
#define I2S_WS       5   // D4
#define I2S_BCK      6   // D5
#define I2S_DATA     43  // D6

const int ntcPins[] = {A0, A1, A2, A3};

#define SAMPLE_RATE    48000
#define BUFFER_SAMPLES 1024

// ========================================
// DATA STRUCTUUR
// ========================================
struct __attribute__((packed)) DataPacket {
    uint16_t syncWord;
    uint8_t packetType;
    uint32_t sequenceNumber;
    union {
        int16_t audio[BUFFER_SAMPLES];
        float temps[4];
    } payload;
};

QueueHandle_t dataQueue = NULL;
uint32_t globalSequenceCount = 0;
```

```
i2s_config_t i2s_config = {
    .mode = (i2s_mode_t)(I2S_MODE_MASTER | I2S_MODE_RX),
    .sample_rate = SAMPLE_RATE,
    .bits_per_sample = I2S_BITS_PER_SAMPLE_32BIT,
    .channel_format = I2S_CHANNEL_FMT_ONLY_LEFT,
    .communication_format = I2S_COMM_FORMAT_I2S,
    .intr_alloc_flags = ESP_INTR_FLAG_LEVEL1,
    .dma_buf_count = 8,
    .dma_buf_len = 512,
    .use_apll = false
};

// ========================================
// NTC FUNCTIES
// ========================================
float Read_Single_NTC(int pin) {
    float Vntc = (analogRead(pin) * 3.3) / 4095.0;
    if (Vntc < 0.05) return 0.0;
    float Rntc = 10000.0 * ((3.3 / Vntc) - 1.0);
    float a = 639.5, b = -0.1332, c = -162.5;
    return a * pow(Rntc, b) + c;
}

float Get_Avg_NTC(int pin) {
    float sum = 0;
    for(int i=0; i<10; i++) {
        sum += Read_Single_NTC(pin);
        delayMicroseconds(100);
    }
    return sum / 10.0;
}

// ========================================
// TAAK 1: AUDIO (CORE 1)
// ========================================
// Deze taak leest de microfoon uit en stopt de data in de wachtrij
void audioTask(void *pvParameters) {
    static int32_t raw_buffer[BUFFER_SAMPLES];
    static DataPacket pkg;
    pkg.syncWord = 0xAAAA;

    while (true) {
        size_t bytes_read = 0;
        i2s_read(I2S_NUM_0, raw_buffer, sizeof(raw_buffer), &bytes_read,
portMAX_DELAY);

        if (bytes_read > 0) {
            pkg.packetType = 0; // 0 = Audio
            pkg.sequenceNumber = globalSequenceCount++;

            int samples_read = bytes_read / 4;
            for (int i = 0; i < samples_read; i++) {
                pkg.payload.audio[i] = (int16_t)(raw_buffer[i] >> 16);
```

```
            }
            // Stop in de wachtrij
            xQueueSend(dataQueue, &pkg, 0);
        }
    }
}

// ========================================
// TAAK 2: MANAGER (CORE 0)
// ========================================
// Deze taak haalt data uit de wachtrij en print het naar Serial
void dataManagerTask(void *pvParameters) {
    static DataPacket rx_pkg;
    static DataPacket temp_pkg;
    unsigned long lastTempMillis = 0;
    temp_pkg.syncWord = 0xAAAA;

    // Teller om serial monitor niet te overspoelen met audio meldingen
    int audioPrintCounter = 0;

    while (true) {
        // --- A. TEMP METING (Elke seconde) ---
        if (millis() - lastTempMillis >= 1000) {
            lastTempMillis = millis();
            temp_pkg.packetType = 1; // 1 = Temp
            temp_pkg.sequenceNumber = globalSequenceCount;
            for(int i=0; i<4; i++) temp_pkg.payload.temps[i] =
Get_Avg_NTC(ntcPins[i]);

            // PRINT NAAR SERIAL MONITOR
            Serial.printf("\n🌡️ TEMP: T1:%.1f | T2:%.1f | T3:%.1f | T4:%.1f\n",
                          temp_pkg.payload.temps[0], temp_pkg.payload.temps[1],
                          temp_pkg.payload.temps[2], temp_pkg.payload.temps[3]);
        }

        // --- B. VERWERK DATA UIT WACHTRIJ ---
        // We wachten maximaal 10ms of er data binnenkomt
        if (xQueueReceive(dataQueue, &rx_pkg, pdMS_TO_TICKS(10))) {

            // We doen niks met de data (geen wifi/sd), maar we printen wel status

            if (rx_pkg.packetType == 0) { // Audio packet
                audioPrintCounter++;
                // Print elke 100 pakketjes even een stipje, anders crasht serial
monitor door snelheid
                if (audioPrintCounter >= 100) {
                    Serial.print("."); // Laat zien dat audio loopt
                    audioPrintCounter = 0;
                }
            }
        }
    }
}
```

```
// ========================================
// SETUP
// ========================================
void setup() {
    Serial.begin(115200);
    analogReadResolution(12);
    pinMode(LED_BUILTIN, OUTPUT);
    delay(2000);

    Serial.println("\n--- START SYSTEEM (ALLEEN SERIAL) ---");

    // Maak de wachtrij in het snelle geheugen (PSRAM)
    dataQueue = xQueueCreateWithCaps(200, sizeof(DataPacket), MALLOC_CAP_SPIRAM);
    if (dataQueue == NULL) { Serial.println("❌ FOUT: Geen PSRAM Queue!");
while(1); }

    // I2S Starten (Microfoon)
    i2s_driver_install(I2S_NUM_0, &i2s_config, 0, NULL);
    i2s_pin_config_t pins = { .bck_io_num = (gpio_num_t)I2S_BCK, .ws_io_num =
(gpio_num_t)I2S_WS, .data_out_num = I2S_PIN_NO_CHANGE, .data_in_num =
(gpio_num_t)I2S_DATA };
    i2s_set_pin(I2S_NUM_0, &pins);

    // Taken starten op de twee processorkernen
    xTaskCreatePinnedToCore(audioTask, "Audio", 20000, NULL, 10, NULL, 1);
    xTaskCreatePinnedToCore(dataManagerTask, "Manager", 20000, NULL, 5, NULL, 0);

    Serial.println("🚀 Meting is gestart! Check Serial Monitor.");
}

void loop() {
    // Knipper LED om te laten zien dat hij niet vastgelopen is
    digitalWrite(LED_BUILTIN, (millis() / 500) % 2);
    vTaskDelay(100);
}
```

18.4 SD writing and no Wi-Fi connection (final code used)

```
#include <driver/i2s.h>
#include <SD.h>
#include <SPI.h>
#include "FS.h"
#include "esp_heap_caps.h"

// ========================================
// CONFIGURATIE (PINNEN)
// ========================================
#define I2S_WS       5   // D4
#define I2S_BCK      6   // D5
```

```
#define I2S_DATA     43  // D6
#define SD_CS        44  // D7 (Chip Select voor SD Kaart)

const int ntcPins[] = {A0, A1, A2, A3};

#define SAMPLE_RATE    48000
#define BUFFER_SAMPLES 1024

// We gebruiken een variabele bestandsnaam
char currentFilename[32] = "/storage0.bin";

// ==========================================
// DATA STRUCTUUR
// ==========================================
struct __attribute__((packed)) DataPacket {
    uint16_t syncWord;
    uint8_t packetType;
    uint32_t sequenceNumber;
    union {
        int16_t audio[BUFFER_SAMPLES];
        float temps[4];
    } payload;
};

QueueHandle_t dataQueue = NULL;
uint32_t globalSequenceCount = 0;

i2s_config_t i2s_config = {
    .mode = (i2s_mode_t)(I2S_MODE_MASTER | I2S_MODE_RX),
    .sample_rate = SAMPLE_RATE,
    .bits_per_sample = I2S_BITS_PER_SAMPLE_32BIT,
    .channel_format = I2S_CHANNEL_FMT_ONLY_LEFT,
    .communication_format = I2S_COMM_FORMAT_I2S,
    .intr_alloc_flags = ESP_INTR_FLAG_LEVEL1,
    .dma_buf_count = 8,
    .dma_buf_len = 512,
    .use_apll = false
};

// ==========================================
// NTC FUNCTIES
// ==========================================
float Read_Single_NTC(int pin) {
    float Vntc = (analogRead(pin) * 3.3) / 4095.0;
    if (Vntc < 0.05) return 0.0;
    float Rntc = 10000.0 * ((3.3 / Vntc) - 1.0);
    float a = 639.5, b = -0.1332, c = -162.5;
    return a * pow(Rntc, b) + c;
}

float Get_Avg_NTC(int pin) {
    float sum = 0;
    for(int i=0; i<10; i++) {
        sum += Read_Single_NTC(pin);
        delayMicroseconds(100);
    }
    return sum / 10.0;
}

// ==========================================
// BESTANDSNAAM KIEZEN
// ==========================================
void determineFilename() {
    int counter = 0;
    while (true) {
        snprintf(currentFilename, 32, "/storage%d.bin", counter);
        if (!SD.exists(currentFilename)) {
            Serial.print("🆕 Nieuw bestand aangemaakt: ");
            Serial.println(currentFilename);
            break;
        }
        counter++;
    }
}

// ==========================================
// TAAK 1: AUDIO (CORE 1)
// ==========================================
void audioTask(void *pvParameters) {
    static int32_t raw_buffer[BUFFER_SAMPLES];
    static DataPacket pkg;
    pkg.syncWord = 0xAAAA;

    while (true) {
        size_t bytes_read = 0;
        i2s_read(I2S_NUM_0, raw_buffer, sizeof(raw_buffer), &bytes_read,
portMAX_DELAY);

        if (bytes_read > 0) {
            pkg.packetType = 0; // 0 = Audio
            pkg.sequenceNumber = globalSequenceCount++;

            int samples_read = bytes_read / 4;
            for (int i = 0; i < samples_read; i++) {
                pkg.payload.audio[i] = (int16_t)(raw_buffer[i] >> 16);
            }
            // Stop in wachtrij voor opslag
            xQueueSend(dataQueue, &pkg, 0);
        }
    }
}

// ==========================================
// TAAK 2: MANAGER & SD WRITER (CORE 0)
// ==========================================
void dataManagerTask(void *pvParameters) {
    static DataPacket rx_pkg;
    static DataPacket temp_pkg;
```

```
    unsigned long lastTempMillis = 0;
    temp_pkg.syncWord = 0xAAAA;

    File dataFile;

    // Open het bestand één keer
    dataFile = SD.open(currentFilename, FILE_WRITE);
    if (!dataFile) {
        Serial.println("FOUT: Kan bestand niet openen!");
        while(1) { digitalWrite(LED_BUILTIN, HIGH); delay(100);
digitalWrite(LED_BUILTIN, LOW); delay(100); }
    }

    int packetsWrittenSinceSave = 0;

    while (true) {
        // --- A. TEMP METING ---
        if (millis() - lastTempMillis >= 1000) {
            lastTempMillis = millis();
            temp_pkg.packetType = 1; // 1 = Temp
            temp_pkg.sequenceNumber = globalSequenceCount;
            for(int i=0; i<4; i++) temp_pkg.payload.temps[i] =
Get_Avg_NTC(ntcPins[i]);

            // Stuur naar queue zodat hij OOK wordt opgeslagen op SD
            xQueueSend(dataQueue, &temp_pkg, 0);

            // Print status
            Serial.printf("REC: %s | Pkts: %d | Temp: %.1f\n",
                          currentFilename, globalSequenceCount,
temp_pkg.payload.temps[0]);
        }

        // --- B. SD SCHRIJVEN ---
        // Haal data uit de wachtrij
        if (xQueueReceive(dataQueue, &rx_pkg, pdMS_TO_TICKS(10))) {

            // Schrijf de ruwe bytes naar de SD kaart
            dataFile.write((uint8_t *)&rx_pkg, sizeof(DataPacket));
            packetsWrittenSinceSave++;

            // Elke 50 pakketjes (~1 keer per seconde) fysiek opslaan
            if (packetsWrittenSinceSave >= 50) {
                dataFile.flush();
                packetsWrittenSinceSave = 0;
                digitalWrite(LED_BUILTIN, !digitalRead(LED_BUILTIN)); // Knipper
LED bij save
            }
        }
    }
}

// ==========================================
// SETUP
```

```
// ==========================================
void setup() {
    Serial.begin(115200);
    analogReadResolution(12);
    pinMode(LED_BUILTIN, OUTPUT);
    delay(2000);

    Serial.println("\n--- START OFFLINE RECORDER ---");

    // Maak Queue in PSRAM
    dataQueue = xQueueCreateWithCaps(200, sizeof(DataPacket), MALLOC_CAP_SPIRAM);
    if (dataQueue == NULL) { Serial.println("FOUT: Geen PSRAM Queue!"); while(1); }

    // Start SD Kaart
    SPI.begin(7, 8, 9, SD_CS);
    if (!SD.begin(SD_CS)) {
        Serial.println("Geen SD kaart gevonden! Stop.");
        while(1);
    }
    Serial.println("SD Kaart OK.");

    // Bepaal bestandsnaam
    determineFilename();

    // I2S Starten
    i2s_driver_install(I2S_NUM_0, &i2s_config, 0, NULL);
    i2s_pin_config_t pins = { .bck_io_num = (gpio_num_t)I2S_BCK, .ws_io_num =
(gpio_num_t)I2S_WS, .data_out_num = I2S_PIN_NO_CHANGE, .data_in_num =
(gpio_num_t)I2S_DATA };
    i2s_set_pin(I2S_NUM_0, &pins);

    // Taken starten
    xTaskCreatePinnedToCore(audioTask, "Audio", 20000, NULL, 10, NULL, 1);
    xTaskCreatePinnedToCore(dataManagerTask, "SD_Writer", 20000, NULL, 5, NULL, 0);

    Serial.println(" Opname gestart! Koppel niet los zolang LED knippert.");
}

void loop() {
    // Loop is leeg, alles gebeurt in taken
    vTaskDelay(1000);
}
```

19 Consent forms

19.1 Panel

U wordt uitgenodigd om deel te nemen aan een onderzoek genaamd Slimme knie wearable voor Artrose patiënten. Dit onderzoek wordt uitgevoerd door Manon van der Stap van de TU Delft. In samenwerking met het Erasmus MC.

Het doel van dit project is het ontwerpen van een wearable die temperatuur en geluid rondom de knie kan detecteren. Het doel van deze test is het testen en valideren van onderdelen van de wearable. Het zal ongeveer 30 minuten in beslag nemen. De data zal gebruikt worden in een afstudeerverslag om concepten te valideren, keuzes te maken, en ook om het ontwerp verder te vormen tijdens het project. U wordt gevraagd om uw mening over de getoonde onderdelen te vertellen. Vragen als, wat heeft uw voorkeur en waarom kunnen worden gesteld.

Het risico van een databreuk is aanwezig. Wij doen ons best om uw antwoorden vertrouwelijk te houden. We minimaliseren de risico's door alle data anoniem te verzamelen, bij het maken van een foto of video worden gezichten (als deze erop staan) wazig gemaakt. Alle bestanden, als tekst, audio, en foto worden op de server van de Tu Delft bewaard.

Uw deelname aan dit onderzoek is volledig vrijwillig, en u kunt zich elk moment terugtrekken zonder reden op te geven. U bent vrij om vragen niet te beantwoorden. Terugtrekken kan tot een week na de deelname, hierna is terugtrekken niet meer mogelijk.

Alvast bedankt voor uw medewerking aan dit project!

19.2 Students, comfort validation

U wordt uitgenodigd om deel te nemen aan een onderzoek genaamd Slimme knie wearable voor Artrose patiënten. Dit onderzoek wordt uitgevoerd door Manon van der Stap van de TU Delft. In samenwerking met het Erasmus MC.

Het doel van dit project is het ontwerpen van een wearable die temperatuur en geluid rondom de knie kan detecteren. Het doel van deze test is het testen en valideren van de wearable. Het zal ongeveer 4 uur dragen en 30 minuten bespreking in beslag nemen. De data zal gebruikt worden in een afstudeerverslag om concepten te valideren, keuzes te maken, en ook om het ontwerp verder te vormen tijdens het project. U wordt gevraagd om de wearable aan te doen, en uw mening erover te vertellen. Vragen als, hoe vindt u hem zitten, hoe gaat het aantrekken, zou je hem de hele dag om kunnen houden, etc. er kan ook worden gevraagd om een cijfer te geven met betrekking tot een categorie bijvoorbeeld comfort, 1 is niet comfortabel en 10 is heel comfortabel.

Het risico van een databreuk is aanwezig. Wij doen ons best om uw antwoorden vertrouwelijk te houden. We minimaliseren de risico's door alle data anoniem te verzamelen, bij het maken van een foto of video worden gezichten (als deze erop staan) wazig gemaakt. Alle bestanden, als tekst, audio, en foto worden op de server van de Tu Delft bewaard.

Uw deelname aan dit onderzoek is volledig vrijwillig, en u kunt zich elk moment terugtrekken zonder reden op te geven. U bent vrij om vragen niet te beantwoorden. Terugtrekken kan tot een week na de deelname, hierna is terugtrekken niet meer mogelijk.

Alvast bedankt voor uw medewerking aan dit project!

19.3 Data validation

U wordt uitgenodigd om deel te nemen aan een onderzoek genaamd Slimme knie wearable voor Artrose patiënten. Dit onderzoek wordt uitgevoerd door Manon van der Stap van de TU Delft. In samenwerking met het Erasmus MC.

Het doel van dit project is het ontwerpen van een wearable die temperatuur en geluid rondom de knie kan detecteren. Het doel van deze test is het testen en valideren van de wearable. Het zal maximaal 20 minuten in beslag nemen. De data zal gebruikt worden in een afstudeerverslag om de data te valideren. U wordt gevraagd om de wearable aan te doen, en oefeningen te doen.

Het risico van een databreuk is aanwezig. Wij doen ons best om uw antwoorden vertrouwelijk te houden. We minimaliseren de risico's door alle data anoniem te verzamelen, bij het maken van een foto of video worden gezichten (als deze erop staan) wazig gemaakt. Alle bestanden, als tekst, audio, en foto worden op de server van de Tu Delft bewaard.

Uw deelname aan dit onderzoek is volledig vrijwillig, en u kunt zich elk moment terugtrekken zonder reden op te geven. U bent vrij om vragen niet te beantwoorden. Terugtrekken kan tot een week na de deelname, hierna is terugtrekken niet meer mogelijk.

Alvast bedankt voor uw medewerking aan dit project!