Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Engineering
Department of Mechanical and Industrial Engineering

Cathrine Bartnes

Integrating Physiological Sensors into an Open Source Multimodal Experimental Setup

Master's thesis in Engineering and ICT
Supervisor: Federico Lozano
Co-supervisor: Martin Steinert
June 2022

NTNU
Norwegian University of
Science and Technology

Cathrine Bartnes

Integrating Physiological Sensors into an Open Source Multimodal Experimental Setup

Master's thesis in Engineering and ICT
Supervisor: Federico Lozano
Co-supervisor: Martin Steinert
June 2022

Norwegian University of Science and Technology
Faculty of Engineering
Department of Mechanical and Industrial Engineering

NTNU | Norwegian University of Science and Technology

Department of Mechanical and Industrial Engineering

Integrating Physiological Sensors into an Open Source Multimodal Experimental Setup

Cathrine Bartnes

Supervisor:
Federico Lozano

Co-supervisor:
Martin Steinert

June, 2022

Abstract

This thesis shows how to develop a multimodal experimental setup, utilizing open-source software. To best develop products and systems, engineers must have an understanding of human responses in interaction with computers and machines. Research challenges today include physiological sensors using their own proprietary software, which make it difficult to run experiments with multiple sensors simultaneously. Lab Streaming Layer (LSL) is one solution, an open-source software distribution that handles data collection and presentation as well as networking and time-synchronization. The work described in this thesis builds on the project assignment where custom code was implemented to send the data stream from the proprietary sensors, Shimmer3 EXG and GSR, to LSL. The experiment setup integrates electrocardiogram (ECG), and galvanic skin response (GSR), together with cognitive measurement tools such as electroencephalography (EEG) and functional near-infrared spectroscopy (fNIRS) and behavioral recording by eye tracking technology, which is tested in a pilot experiment.

What the sensors measure and how they measure it, as well as the technologies used, are all thoroughly explained. The pilot experiment is an extension of previous work carried out by Goucher-Lambert et al. (2019), where participants are asked to develop ideas based on different questions and varying words of inspiration. The Ideation experiment is implemented in PsychoPy, open-source software for behavioral science experiments. The motivation for conducting this experiment is to replicate and validate the work that has been done, in addition to adding new information to the field. As it is claimed that we are in a replication crisis, it is critical to confirm earlier findings in order to maintain high-quality research. Further, the thesis explains how the data streams should be processed, and the results are presented from the pilot experiment. The results show that the experiment setup can be used by known open-source processing tools, which indicates that it is within the standard of relevant scientific research. The setup is able to both record and process data, with which advanced analyzes can be performed. A demonstration of the setup in a real-world situation shows flexibility and mobility, which is desirable in the field of engineering.

Currently, the selection of multimodal and mobile experimental setups is minimal, which indicates it is a need for the setup presented in this thesis. As technology becomes more complicated and is used more frequently in everyday settings, it is essential to understand how humans interact with and react to computers and machines in the context in which they are used.

The work completed for this thesis resulted in a paper that was accepted for publication at the NordDesign conference in 2022.

Sammendrag

Denne oppgaven viser hvordan man kan utvikle et multimodalt eksperiment oppsett, ved hjelp av open-source programvare. Det er viktig for ingeniører å vite hvordan et menneske reagerer i interaksjon med maskiner og datamaskiner for å utvikle produkter og systemer best mulig. Utfordringen mange forskere står overfor i dag er at fysiologiske sensorer gjerne kommer med sin egen programvare. Dette gjør det vanskelig å utføre eksperiment med flere sensorer samtidig. Dette har blitt løst ved distribusjonen Lab Streaming Layer (LSL), som er en open-source distribusjon. Denne håndterer alt fra tilkobling av flere sensorer, til synkronisering og opptak av dataene fra disse sensorene. Arbeidet bygger videre fra prosjektoppgaven der det ble implementert egendefinert kode for å sende datastrømmen fra de proprietære sensorene, Shimmer3 EXG og GSR, til LSL. Eksperimentoppsettet integrerer elektrokariogram (EKG), galvanisk hudledeevne ("galvanic skin response" på engelsk, forkortet GSR), sammen med kognitive måleverktøy som elektroencefalografi (EEG) og funksjonell nær-infrarød spektroskopi (fNIRS) og atferdsmessig opptak ved blikksporingsteknologi, som blir testet i et pilot eksperiment.

Det er gitt en grundig forklaring på hva sensorene måler og hvordan de måler det, i tillegg til hvilke teknologier som blir brukt. Pilot eksperimentet er en utvidelse av tidligere arbeid utført av Goucher-Lambert et al. (2019), der deltagere skal utvikle ideer basert på tolv ulike spørsmål og varierende inspirasjonsord. Ideerings eksperimentet er implementert i PsychoPy, en open-source programvare for atferdsvitenskapelige eksperimenter. Motivasjonen for å utføre nettopp dette eksperimentet er for å replisere og validere arbeidet som har blitt gjort, i tillegg til å tilføre ny informasjon til feltet. I og med at det hevdes at vi er i en replikasjonskrise, er det avgjørende å bekrefte tidligere funn for å opprettholde forskning av høy kvalitet. Videre forklares det hvordan datastrømmene burde prosesseres, og resultatene funnet ved pilot eksperimentet. Resultatet viser at oppsettet kan brukes av kjente open-source prosesseringsverktøy, som indikerer at oppsettet er innenfor standarden innen relevant vitenskapelig forskning. Den klarer både å ta opp og prossesere data, som det videre kan gjøres avanserte analyser med. En demosntrasjon av oppsettet i en virkelig situasjon, viser fleksibiliteten og mobiliteten, noe som er ønskelig innen ingeniør feltet.

Foreløpig er utvalget av multimodale og mobile eksperimentoppsett minimale, noe som gjenspeiler at behovet for eksperimentoppsettet beskrevet i denne oppgaven er stort. Ettersom teknologien blir mer komplisert og brukes oftere i hverdagsmiljøer, er det viktig å forstå hvordan mennesker interagerer med og reagerer på datamaskiner og maskiner i konteksten de brukes i.

Arbeidet som ble gjort i sammenheng med denne oppgaven har resultert i en artikkel, akseptert for NordDesign konferansen 2022.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

GSR	Galvanic Skin Response
ECG	Electrocardiogram
EEG	Electroencephalography
fNIRS	Functional Near-Infrared Spectroscopy
LD	Near-Infrared
LSL	Lab Streaming Layer
XDF	Extensible Data Format
HCI	Human-Computer Interaction
HMI	Human-Machine Interaction
HR	Heart Rate
HRV	Heart Rate Variability
bpm	Beats per Minute
fMRI	Functional Magnetic Resonance Imaging
SCR	Skin Conductance Response
SCL	Skin Conductance Level
ERP	Event Related Potential

Chapter 1

Introduction

Within engineering and interaction design, physiological sensors can give valuable information on how human emotion affects behavior. To evaluate human cognition in interaction with technical systems and user interfaces, experiments with physiology sensors should be conducted (Balters & Steinert, 2017). Currently, most experiments with physiology and neuroimaging measurements are conducted within laboratory settings (Goucher-Lambert et al., 2019; Leikanger, 2016). The trade-off between empirical control and ecological validity must be addressed (Hay et al., 2020). Because in situ experiments are carried out in the desired context, they have high ecological validity. As technology is often used in unpredictable, real-world environments, such studies will be beneficial to engineers both at the early design stage and during the evaluation stage of existing systems (Dybvik, Kuster Erichsen, et al., 2021). In order to account for the unforeseen factors in an in situ study, multiple sources of measurements are needed to assure that the reaction is, in fact, a reaction to the given stimuli. Combining neuroimaging modalities with systemic- and behavior measures enables data and method triangulation, which increases the validity of results and reduces bias and error (Balters & Steinert, 2017; Steinert & Jablokow, 2013). Data triangulation is defined according to Blessing and Chakrabarti (2009) as "the combination of multiple data sources and research methods, application of different theoretical perspectives, and use of multiple observers to reduce or at least detect bias and error."

The experiment setup described in this thesis is a mobile multimodal physiological sensory experiment that utilizes open-source distributions. The setup is an extension on previous work from TrollLABS, combining a wearable electroencephalography (EEG) and functional near-infrared spectroscopy (fNIRS) sensor setup (Dybvik, Kuster Erichsen, et al., 2021), eye-tracking technology (Abelson, 2021), Ideation experiment (Goucher-Lambert et al., 2019), and electrocardiography (ECG) and galvanic skin response GSR into one single experiment setup, utilizing the open-source distribution LSL (Kothe et al., 2019b) and PsychoPy (Peirce et al., 2019), a software for behavioral science experiments.

1.1 Problem description

The aim of the project described in this thesis is to develop a low-cost multimodal sensor setup, consisting of EEG, fNIRS, ECG, GSR, and eye-tracking, with the use of open-source software distributions. The setup should provide cognitive activity measurements through fNIRS and EEG, systemic measures via ECG and GSR, as well as behavior measurements through eye-tracking. A pilot experiment, in addition to a demonstration of the setup in situ, is conducted to ensure the setup is fully functional and can be used to execute statistical analysis which corresponds to other findings reported from relevant literature. Additionally, the project should follow NTNU's policy of open science [1], by keeping research processes open and transparent, as well as making the research results available.

1.2 Thesis scope and limitations

The thesis describes a multimodal sensory experiment setup, including the sensors ECG, GSR, EEG, fNIRS, and Eye-tracking. The setup has been tested and successfully executed with all the sensors running simultaneously. The experiment is low cost and highly reproducible thanks to the open-source distribution Lab Streaming Layer (LSL) and PsychoPy, a software for behavioral science experiments, in addition to the analysis tools, NeuralKit2 and MNE.

Since all the sensors collect and process the data differently, a data analysis section is included to inform how to read the data output from the sensors. Even though LabRecorder manages the sensors and synchronizes the timestamps, the process of gathering the sensory input is handled differently by the sensors before LabRecorder captures it, such as the sampling rate and the human response that is being recorded.

Included is a paper accepted for the NordDesign 2022 conference. The paper discusses the importance of open-source and how the research domain should incorporate the philosophy to take advantage of the many effects it has.

A demonstration of how to use the various analysis software's is given in the results section, however, a thorough analysis of the data is outside the scope of this thesis.

1.3 Thesis structure

The thesis background and motivation are presented first, in Section 2. Here is a description of previous work related to the thesis described, in addition to the motivation for conducting the Ideation experiment. Secondly, Section 3 explains the various modules in the setup, from what the sensors measure to the software used. Section 4 presents the paper accepted to the NordDesign 2022 conference. Then, in Section 5, a detailed run-through of the pilot experiment is explained, before Section 6, where a data processing section is included to explain how to read all the sensor data in order to draw the right conclusions from the comprehensive sensor data output. The

[1] https://www.ntnu.edu/policy-for-open-science

results in Section 7, from the Ideation pilot experiment, are then presented, with the use of various analysis tools. Section 8 presents a demonstration of the experiment setup in situ. Then there is a discussion of the results and setup's usability in addition to arguments about whether the setup contributes to the research field in Section 9. Finally, there are some concluding remarks and thoughts about further work.

Chapter 2

Project Background

The human response can be captured through physiological sensors. The sensors have traditionally been used within the field of psychology, neuroscience, and cognitive science, to understand how human emotion can affect behavior. That knowledge is valuable within the engineering perspective as well, given the increasing interaction with engineering systems (Balters & Steinert, 2017). Human-computer interaction (HCI) is a multidisciplinary subject, including the fields of computer science, cognitive science, and product development (Dix et al., 2004). The need to understand the human reaction to stimuli is critical, as inadequate attention to user interaction can put entire systems at risk.

To contribute to the field, this thesis will present an open-source multimodal physiology sensor experiment setup. The thesis is a part of ongoing research at TrollLABS, a lab that aims to combine physiological sensors with engineering design to improve the development of new products. Previously there have been conducted experiments with various physiological sensors. These include EEG, fNIRS, GSR, ECG, and eye-tracking. However, never before have all sensors been integrated into the same experiment setup. In addition to expanding previous experiment setups, the proprietary software program used to conduct the experiments have been replaced with an open-source alternative called LabRecorder, making the setup low-cost. LabRecorder is the default recording program released with the LSL distribution. Utilizing LSL simplifies the data collection process, as all input is synchronized and exported into one single file.

The majority of physiology and neuroimaging experiments are being carried out in laboratories (Goucher-Lambert et al., 2019; Leikanger, 2016), making it hard to obtain ecological validity of the findings. In situ experiments holds a much higher ecological validity, the downside being experimental control. In order to achieve good results, there are multiple factors to take into account. One is variables changing due to reasons other than stimuli. To ensure the response is caused by the stimuli and not random variables, multiple measures to the same stimuli must be performed. Data triangulation increases the validity of the results and reduces bias and error (Dybvik, Kuster Erichsen, et al., 2021). All sensors used in the experiment setup are relatively non-constraining and mobile, allowing for in situ experiments, as will be demonstrated in Section 8.

The value of combining sensors is great. Both eye-tracking, EEG and fNIRS are widely used as stand-alone methods (Abelson, 2021; Dybvik, Kuster Erichsen, et al., 2021). However, they can be much more powerful when combined with other technologies. Eye-tracking provides a record of where a person is focusing their visual attention while EEG and fNIRS provide a record of how the brain responds to it, giving a more complete picture of the response (Carter & Luke, 2020). The complementary technologies add significantly ecological validity by allowing participants to move their eyes freely.

The thesis contributes to the open research community by releasing all source code and related findings for everyone to see. Even though there is a lack of incentives to be transparent within science (Nosek et al., 2015), the benefits of doing so should convince other researchers to contribute to open research. Research with an open-source approach offers the same innovation and rapidity that open-source development does. Due to the well-documented and executed experiment setups executed at TrollLABS, the setup has been expanded by incorporating more physiology sensors into it. The new features presented in this thesis take advantage of previous work, bringing the setup a step closer to a fully mobile multimodal experiment setup to use in situ.

2.1 Findings from the original study

Little is known about the neurological process that supports design cognition involving inspirational stimuli, including analogies (Goucher-Lambert et al., 2019). A study done by Goucher-Lambert et al. (2019) used functional magnetic resonance imaging (fMRI) to investigate design concept generation with and without the support of inspirational stimuli. The goal of the study was to determine which unique areas of the brain are involved during concept generation, in addition to areas involved when reasoning about inspirational stimuli that are either close to or distant from the problem domain. Gaining insight into the neural process of cognition will improve how current inspirational stimuli are used to impact design problem-solving strategies. Consequently, the findings will affect and aid the development of theories, methods, and tools that support the creative potential of designers (Goucher-Lambert et al., 2019).

The results from the original study suggest there are two different types of solution strategies. The *inspired internal search* involves inspirational stimuli where the participant recognizes the meaning and makes connections with retrieved concepts from memory in order to create new ideas (Goucher-Lambert et al., 2019). *Unsuccessful external search* excludes inspirational stimuli, increasing the activity in the primary visual processing-related brain regions, indicating participants continue to search the design problem space for clues and insight (Goucher-Lambert et al., 2019). The results imply that in order to successfully generate ideas, it is beneficial to have inspirational stimuli.

The study has the limitation of reverse inference conclusions, which negatively impacts the interpretation of the collected neuroimaging data (Goucher-Lambert et al., 2019). Reverse inference reasons backward, explaining unexpected brain activation by referring to other studies that found activation in the same region (Poldrack, 2006). By executing the ideation experiment again, a hypothesis can be set a priori, avoiding the error of reverse inference. The experiment setup is extended with multiple physiological sensors, but the questionnaire is identical.

2.2 The replication crisis

The motivation to replicate the Goucher-Lambert et al. (2019) study, is to validate the findings. Currently, there is a claim of a replication crisis, where significant findings have been replicated less often than expected (Shrout & Rodgers, 2018). It is important to have confidence in the published results, given the influence they will have on future research. As a means to reduce the possibility of faulty results, it is recommended that researchers adopt open science conventions of preregistration and full disclosure (Shrout & Rodgers, 2018). Open science encourages the disclosure of predictions, analysis plans, data, and supplementary material to the general scientific community. Today there is few incentives to double-check the continuous contribution of scientific papers published, especially because it is expected to produce big results frequently. Hence, Shrout and Rodgers (2018) recommend power analyses should be developed with more sophistication, effects sizes should be considered carefully, and that nonsignificant findings as well as statistically significant should be disclosed in detail. The paper included in this thesis explains how open-source technology offers rapidity and innovation opportunities, and how open research can learn from computer science about the advantages.

Even though the experiment presented in this thesis is not a replication of the Goucher-Lambert et al. (2019) study, but rather an extension, it can be considered as a way to add additional information to the findings from the original study. The intention is to replicate the findings with the use of other physiology sensors.

Chapter 3

Theory and technology

The theory chapter describes the various sensors and technologies used in the experiment setup. A description of what each sensor measures, how it measures it, and how it has been used in previous research is provided, in addition to the various software and analysis tools used.

3.1 Sensors

3.1.1 EEG

Electroencephalography (EEG) detects electrical brain activity using electrodes attached to the scalp. Local current flow is caused by the activation of neurons (brain cells), between the potential signal electrode and the corresponding potential electrode. The electrical activity can only be measured on the head surface if enough neurons in the same area fire simultaneously (Balters & Steinert, 2017).

The EEG signal is made up of oscillations of different frequencies, which indicates different states of mind. The change in brain wave rhythms are measurable and are correlated to specific brain activities, such as perception, movement, sensory registration, and cognitive processes related to attention, learning, and memory (Başar et al., 2000). Previously, neurocognitive EEG research focused on Event-Related-Potential (ERP) indices, which reflect the brain's response to certain events. The ERP indices are calculated by averaging the continuous EEG signal over many trials to cancel out oscillatory background activity, which was considered noisy (Antonenko et al., 2010). Today, however, the attention is yet again on the dynamics of brain oscillations (Antonenko et al., 2010; Balters & Steinert, 2017), and how the neural activity responds to well-defined events (Başar et al., 2000). The electrical activity in the brain generates five distinct frequencies, ranging from delta (δ): 0.5-4 Hz, theta (θ): 4-8 Hz, alpha (α): 8-12 Hz, beta (β) : 12-35 Hz, to gamma (γ): >35 Hz (Abhang et al., 2016). Both delta and theta waves appear in an unconscious state whereas alpha waves are present during a relaxed but conscious state, with the eyes closed (Balters & Steinert, 2017). Beta waves are present when the eyes are open and the attention is active, and gamma represents deep concentration. It is important to be aware that various regions of the brain do not release the same frequency of brain waves at the same time.

EEG is a non-invasive, portable, high temporal resolution, and relatively inexpensive sensor, however, it has its limitations. Low spatial resolution, proneness to noise from motion, and artifacts due to sweat and/or muscle activity are all drawbacks of EEG (Dybvik, Kuster Erichsen, et al., 2021).

EEG is a popular tool for diagnosis and management of patients with seizure disorders(Smith, 2005), such as encephalopathies (metabolic, infectious, degenerative) and focal brain lesions (cerebral infarction, hemorrhage, neoplasms) (Flink et al., 2002). Although epilepsy is primarily diagnosed clinically, electroencephalography (EEG) plays a major role in evaluating it. When epileptic activity is detected, an EEG recording can be used to identify the type of seizure or epilepsy syndrome (Flink et al., 2002).

3.1.2 fNIRS

Functional Near-Infrared Spectroscopy (fNIRS) measures the hemodynamic response, which is the increase and decrease of oxygen in activated brain regions. The measurements are carried out by transmitting near-infrared (NIR) light onto the scalp, where the light has to travel through several layers (the scalp skin, skull, cerebrospinal fluid) before reaching neural tissue where it is absorbed by the hemoglobin. In addition to being absorbed, the light is scattered, which leads to light attenuation. Light attenuation changes depending on the concentration of oxygenated and deoxygenated hemoglobin, which changes based on the intensity of measurements of the input and output light. This can be calculated through the modified Beer-Lambert Law (Pinti et al., 2020). The hemodynamic response is related to when the brain is involved in the execution of certain tasks and needs to meet the increased metabolic demand of the brain. The hemodynamic response reaches a peak 5/6 seconds after the stimulus event. After stimulus onset, it goes back to baseline after approximately a 16 seconds delay. Depending on brain regions, task types and design, and participants' age, the response dynamics can vary (e.g., peak latency, undershoot latency, duration) (Pinti et al., 2020).

Advantages of fNIRS include non-invasiveness, portability, lightweight, compactness, and low cost, in addition to offering a better spatial resolution than EEG. It is also more robust to motion artifacts. The limitations include low temporal resolution, sensitiveness to changes in scalp blood flow unrelated to brain activation, ambient light, and changes to systemic physiology (Dybvik, Kuster Erichsen, et al., 2021).

The key feature of fNIRS is that it presents the opportunity to study relations between brain activity and freely moving participants in everyday life situations (Pinti et al., 2020). Proof-of-concept experiments where participants practice sport, play a musical instrument, and perform daily activities or attention-demanding tasks are conducted to show the flexibility and robustness of fNIRS (Atsumori et al., 2010; Balardin et al., 2017). Even a quantitative comparison of fNIRS and fMRI suggests that fNIRS can be an appropriate substitute for fMRI for studying brain activity related to cognitive tasks (Cui et al., 2011).

3.1.3 ECG

The heart rate is determined using electrocardiography (ECG), which measures the cardiac potential difference in the heart. The electrical potential is determined by placing electrodes on the body's surface and measuring the lead - the difference between them. During each cardiac cycle (heartbeat) these electrodes measure the small electrical changes that result from cardiac muscle depolarization and repolarization (Balters & Steinert, 2017). As a response to internal or external triggers, both the sympathetic and parasympathetic nervous systems can adjust the heart rhythm (Balters & Steinert, 2017).

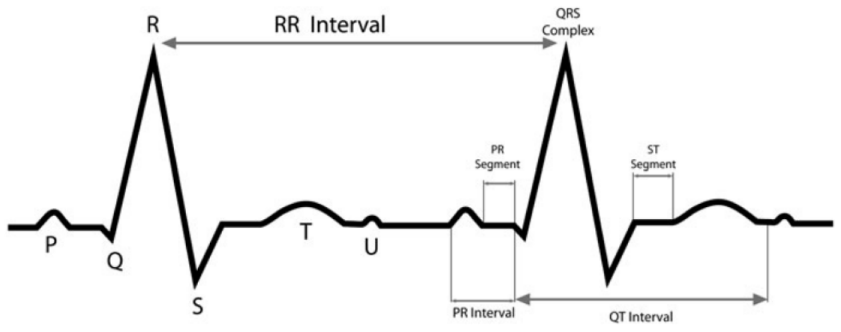


Figure 3.1: The ECG waves, segments, and intervals

The ECG waveform, Figure 3.1, consists of three elements: the isoelectric line (baseline), segments, and intervals. Segments are the duration of the isoelectric line, where there is no electrical activity on the ECG between waves. Intervals are the time between identical segments of neighboring waves (Gacek & Pedrycz, 2011). The P-wave is the first deflection of the ECG and is a result of the depolarization of the atria. The QRS complex corresponds to the ventricular depolarization, that is the contraction of the ventricles. Due to the fact that the ventricles are significantly larger than the atria, the effect of depolarizing the cells in the ventricles is stronger than the effect of the P wave. The repolarization of the ventricles is represented by the succeeding T wave (Balters & Steinert, 2017). In some cases, a U-wave can be seen, which has the same polarity as the preceding T-wave (Gacek & Pedrycz, 2011). The interval between two consecutive cardiac cycles (heartbeats), named the RR interval, can be used to calculate heart rate (HR), a measurement of beats per minute (bpm), and heart rate variability (HRV), a measure of changes in heartbeat frequency.

The application areas of ECG include heart rate monitoring, detection of arrhythmia and coronary artery disease, medication and drug monitoring, as well as cardiac stress testing (Malmivuo & Plonsey, 1995; Pourmohammadi & Maleki, 2020).

3.1.4 GSR

The Galvanic Skin Response (GSR) sensor applies a small voltage to two electrodes to measure the electrical conductance on the skin's surface (Balters & Steinert, 2017). The GSR is caused by sweat glands in the skin being activated by the sympathetic autonomic nervous system. When humans are emotionally stimulated or encounter stimuli, sweat glands stimulate the production of moisture through the skin's pores. It is the change in the balance of positive and negative ions in the secreted fluid that is being measured (iMotions, 2017), which can be expressed as electrical resistance or conductivity. The amplitude of a typical signal is in the range of 0.2-1μS (or, expressed as resistance:1-5 MΩ) (Balters & Steinert, 2017).

Being non-invasive and easy to obtain, the measurement method is widely applied in clinical research (Benedek & Kaernbach, 2010). A lot of information about our unconscious behavior is provided by our skin. For example, the number of fluctuations of skin conductance per second correlates with postoperative pain (Ledowski et al., 2007). In addition, a GSR sensor detects pain fast and continuously, specific to the individual (Storm, 2008). Other fields encompasses schizophrenia (Raine et al., 1999), and epilepsy. People with epilepsy can use biofeedback training to learn how to deliberately manage typically autonomous physiological signals (Nagai et al., 2019).

3.1.5 Eye-tracking

An eye tracker records movement and in what order the gaze is directed. Given the anatomy of the eye, which limits the high acuity vision to a small portion of the visual field, the gaze must move in such a way that the fovea is pointed at the stimuli that are being processed (Carter & Luke, 2020). Due to the eye-mind link, eye tracking can be used to investigate what factors influence visual attention allocation (Just & Carpenter, n.d.). Most eye trackers are video-based and by shining infrared light into the eye, the eye-tracking software detects the reflection of this light on the cornea (Carter & Luke, 2020). The corneal reflection and pupil center are localized during calibration so that the software can determine the gaze position.

Eye-tracking has a wide range of applications, from neuroscience and psychology to industrial engineering and computer science (Carter & Luke, 2020). Since the discovery of eye movement facts such as saccadic suppression, saccade latency, and size of the perceptual span, the eye-tracking research field has moved from a basic science focus to one that emphasizes the application of these facts (Duchowski, 2002). Since eye-tracking enables a deeper understanding of cognitive processes that are not directly related to attention, such as perception, memory, language, and decision making, most research studying mental processes can benefit from it (Carter & Luke, 2020). In addition, the development of better and more adaptable methods of eye-tracking has resulted in a huge increase in the use of eye-trackers.

3.2 Software applications

3.2.1 PsychoPy

PsychoPy is an open-source Python application for creating behavioral research experiments with exact spatial control and stimulus timing (Peirce et al., 2019). The application allows for experiments to be graphically constructed in the Builder interface in addition to the insertion of Python code for maximal flexibility. The latest version of PsychoPy includes support for Pupil Lab's eye tracker, meaning it can calibrate, document, and record eye tracking data without using the designated software developed by Pupil Labs. Obtaining eye-tracking data on a screen experiment should be trivial.

3.2.2 Lab Streaming Layer

The lab streaming layer (LSL) is a platform for the centralized collection and viewing of measurement time series in research experiments that enables both networking, time-synchronization, and (near-) real-time access as well as optionally centralized data collection, viewing, and disk recording (Kothe et al., 2019b).

The distribution includes a core library called "liblsl", a recording program called LabRecorder, file importers, and a variety of applications tailored to various physiology sensors. LSL allows multiple software components with integrated LSL functionality to stream data over the local network and record it with LabRecorder. For high-quality measurements, the library implements several reliability mechanisms, such as transport in accordance with Transmission Control Protocol (TCP), failure recovery, buffered data, and type conversion support (Kothe et al., 2019a).

LabRecorder

LabRecorder is the default recording program that's included with LSL. The software records all available streams in the network into a single file provided with the Extensible Data Format (XDF), synchronizing times between streams ("Overview", 2021). XDF is tailored towards biosignal data and data with a high number of channels. The format is a general-purpose container to handle multi-channel time series with extensive associated meta-information ("Extensible Data Format (XDF)", 2021).

3.2.3 Analysis software

There are several options in how to analyze the recorded data from the various sensors. The Open-source alternatives utilized in this thesis are Nerokit2 (Makowski et al., 2021), MNE (Gramfort et al., 2013), and Pupil Player (Labs, 2022). Neurokit2 is used to analyze the systemic measurements captured from ECG and GSR. The cognitive measurements, fNIRS and EEG, are handled by the MNE software, and lastly, the behavioral measurements recorded by the eye-tracker are visualized by Pupil Lab's software Pupil Player.

NeuroKit2

NeuroKit2 is an open-source Python package for neurophysiological signal processing. The package offers a comprehensive set of data processing routines for a wide range of bodily signals, including ECG and GSR. It facilitates easy data processing methods, through high-level functions and validated pipelines. Furthermore, the package comes with tools for specific processing steps like filtering methods and rate extraction, providing a balance of high-level accessibility and fine-tuned control (Makowski et al., 2021).

MNE

MNE is an open-source Python package for exploring, visualizing, and analyzing cognitive measurement data, such as EEG and fNIRS. Characterizing and locating neural activation in the brain is a challenge that requires expertise in physics, signal processing, statistics, and numerical methods. The package addresses these challenges and provides state-of-the-art algorithms that cover multiple methods of data preprocessing, source localization, statistical analysis, and estimation of functional connectivity between distributed brain regions (Gramfort et al., 2013).

Pupil Player

The Pupil Player provides a range of tools to analyze the recorded data from the eye-tracker. These include blink, and fixation detection, post-hoc pupil detection, post-hoc gaze calibration, and pupil and gaze recordings [1]

[1] https://docs.pupil-labs.com/core/software/pupil-player/

Chapter 4

Development Process

This chapter includes an article accepted for the NordDesign Conference 2022 in Copenhagen, Denmark. The article explains the development process of how to incorporate the various sensors into the open-source software distribution LSL. Traditionally each sensor has its own proprietary software, making it difficult and costly to conduct multimodal experiments. The article gives an example of how to enable proprietary hardware to utilize open-source software. The setup records, synchronizes, and exports high-quality data from multiple sensory measures. Conclusively, a discussion on how the use of open-source resources can benefit the research domain is presented. Open research has the same potential as open-source development in terms of rapidity and innovation, leading to new advances that would not have been possible in closed environments.

NordDesign 2022
August 16-19, 2022
Kgs. Lyngby, Denmark

Promoting and demonstrating open research by adapting proprietary hardware to open-source software

Cathrine Bartnes[1], Agnes Ockernahl[2], Henrikke Dybvik[3] , Martin Steinert[4]

[1]The Norwegian University of Science and Technology (NTNU), Norway
cathrbar@stud.ntnu.no
[2] The Norwegian University of Science and Technology (NTNU), Norway
amockern@stud.ntnu.no
[3] The Norwegian University of Science and Technology (NTNU), Norway
henrikke.dybvik@ntnu.no
[4] The Norwegian University of Science and Technology (NTNU), Norway
martin.steinert@ntnu.no

Abstract

Physiological sensors can give valuable information about the human response when interacting with engineering products or systems, which is essential for engineers to know when developing new innovations. Observing multiple reactions to the same stimuli will help researchers gain a better understanding of how the human body responds. However, there is limited availability of multimodal physiological sensor setups. This article presents an alternative approach utilizing the open-source distribution Lab Streaming Layer (LSL) which records, synchronizes, and exports high-quality data from multiple sensory measures. The challenge is to adapt proprietary hardware to utilize LSL, as most sensors are pre-programmed to interact with their own software. The aim is to reduce the cost associated with subscriptions to proprietary software, by switching to the open-source alternative which promotes higher quality, greater reliability, greater flexibility, and lower cost.

The article presents a case where the proprietary sensors, electrocardiogram (ECG) and galvanic skin response (GSR) provided by Shimmer, adapt to the LSL network. Given the multidisciplinary aspect of the case both Wayfaring, Extreme Programming (XP), and Scrum were used as development methodologies. A description is provided of the software architecture for collecting data from the two Shimmer sensors and how the signals are transmitted to the LSL network.

In addition, the article encourages the use of open-source resources and discusses the benefits that open research will bring. Open research offers similar potential as open-source development in terms of rapidity and innovation. The advantage of allowing researchers to benefit from other researchers' findings and results, leads to advances that would not have been possible in a closed, or proprietary, environment. In light of the insights acquired in the development process, questions regarding companies' attitudes toward open research are addressed. Moreover, the article adds to the field of open research by broadening the possibilities for experimental setups facilitating the acquisition of new knowledge. A demonstration of how open platforms might be utilized to lower the expenses of such research, with the primary aim of promoting the benefits of open-source and how it may benefit open research.

Keywords: Open-source, Open research, LSL, Multimodal experiments, Physiology sensors

1 Introduction

Most physiological sensors are pre-programmed to only interact with their proprietary software, making conducting experiments with multiple physiological sensors time-consuming and expensive. In addition to making it difficult to synchronize sensory data from various sensors across several platforms, the cost of leasing proprietary software can be significant in the long run. This article discusses how the use of open-source resources, namely Lab Streaming Layer (LSL) can benefit the research domain and reduce cost by removing the use of proprietary software. The advantages and drawbacks of open-source software are addressed, demonstrated by a development case where proprietary hardware is adapted to open-source software. The case presents how the proprietary physiological sensors, developed by Shimmer Research (Shimmer Research, n.d.-b), are integrated into the open LSL network. This incorporation of open-source software solutions contributes to a low-cost experiment setup.

The case is inherently multidisciplinary, combining engineering, computer science, signal processing, and physiology. Thus, the development process had to embrace agile development which is influenced by the methodologies of Wayfaring (Gerstenberg et al., 2015; Steinert & Leifer, 2012), Extreme Programming (Sommerville, 2011), and Scrum (Schwaber, 1997).
Within engineering design, the case illustrates how both software and product development overlap.

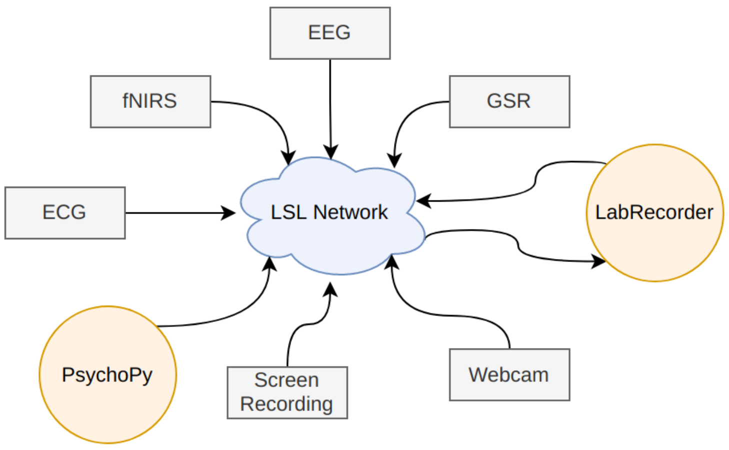
Open research offers the same potential as open-source development in terms of rapidity and innovation. Even though science's credibility would benefit if everyone was more transparent, many individual researchers lack strong incentives to be more transparent (Nosek et al., 2015). Using other researchers’ findings and results allows for further innovation and lead to advances that would not have been possible in a closed environment. Keeping the research transparent and available will contribute to reproducibility and validation. In the same way open-source has its set of terms to fulfil (Open Source Initiative, n.d.-b), the Transparency and Openness Promotion (TOP) Committee has developed shared standards for open practices to change the current incentives to drive researchers toward more openness (Nosek et al., 2015).

2 Case presentation and theoretical background

The case presented in this article takes advantage of the already existing experiment setup (Dybvik et al., 2021), and proposes a solution where it extends the current functionality. Physiological sensors can give valuable information on the human response in interaction with machines and computers (Balters & Steinert, 2017). To gain a better understanding of the human response pattern, it is encouraged to observe multiple physiological responses to the same stimuli. Applying data triangulation, which combines multiple data sources and research methods, will contribute to reducing or at least detecting bias and error (Blessing & Chakrabarti, 2009). Thus, the results will be seen as more credible since each response confirms the other, ensuring that the correct conclusions are drawn. The full experiment setup currently consists of the sensors electroencephalography (EEG), functional near-infrared spectroscopy (fNIRS), galvanic skin response (GSR), and electrocardiogram (ECG), which are recorded by proprietary software. Given the existing experiment setup already has enabled streaming of data to the LSL network, it is only the GSR and ECG sensor that must be adapted to the open-source software distribution LSL. The overarched aim is to collect all measurement data solely by utilizing open-source software. As such, what remains is to implement a custom script to adapt the configuration of GSR and ECG, so the data can be streamed to the LSL network. The case presented here is the development process of retrieving and collecting GSR and ECG data, collected by Shimmer hardware units, and enabling LSL compatibility.

To get a full understanding of the experiment setup and the integration of the various sensors, the necessary technical information is provided. “Figure 1 – Architecture of experiment setup” shows how the LSL network is connected to hardware (represented by boxes) and open-source software (shown by circles).

Figure 1 – Architecture of experiment setup



2.1 Lab Streaming Layer

The Lab Streaming Layer is an open-source system developed by Christian Kothe, David Medine, Chadwick Boulay, Matthew Grivich, and Tristan Stenner. The system handles time-synchronization, networking, and real-time access in addition to viewing and disk recording of the data (Kothe et al., 2019). The system comes with a core library called “liblsl,” a recording program called LabRecorder, file importers, and a wide variety of applications suited for various measurement units. LabRecorder collects all sensor streams available to the LSL network into one single file provided with the Extensible Data Format (XDF)(*Extensible Data Format (XDF)*, 2015/2021). The XDF format is designed specifically for bio signal data, given the general-purpose container format for multi-channel time-series data with extensive associated meta-information.

2.2 Shimmer

Shimmer Research is a leading provider of wearable sensor products including technology services (Shimmer Research, n.d.-b). They provide a proprietary software solution called

ConsensysPRO, which records and displays sensor measurements, in addition to various APIs. The physiological sensors used in the experimental setup described in this article are Shimmer3 GSR+ and EXG. The units come pre-programmed with the LogAndStream firmware, which is the configuration allowing for serial connection through Bluetooth as well as capturing data onto the SD card of the units (Shimmer Research, 2018). The firmware is developed in the proprietary integrated development environment Code Composer Studio (CCS) by Texas Instruments, which is the distributor of the microcontroller used by the Shimmer3 units (Shimmer Research, 2017). The IDE supports C and C++, which is the programming language used to configure the units (Texas Instruments, n.d.-b). Shimmer openly provides a wide range of source code, including the LogAndStream firmware, available through their GitHub account.

2.2.1 GSR

A Galvanic Skin Response (GSR) sensors measure electrodermal activity (EDA), which is the automatic activation of the sweat glands in the skin, due to emotional arousal or the introduction of a new stimulus (Balters & Steinert, 2017; Shimmer Research, n.d.-a).

The Shimmer3 GSR+ unit is equipped with a Texas Instruments MSP430F5437A microcontroller, which among other features consists of a 16-bit RISC architecture and a 12-bit analogue-to-digital converter (ADC) (Texas Instruments, n.d.-c). The mentioned features determine the data signal type, u12, and u16, which must be provided when extracting the data from the sensors with the Struct package.

2.2.2 ECG

An electrocardiography (ECG) sensor measures the electrical activity of the heart. Electrodes are placed on the body's surface and the lead, the difference between them, is measured to determine the electrical potential. Both the sympathetic- and the parasympathetic nervous system can adapt the heart rhythm as a reaction to internal or external triggers (Balters & Steinert, 2017).

The Shimmer3 unit is equipped with two ADS1292R chips from Texas Instruments, which is a 24-bit ADC with integrated respiration impedance (Texas Instruments, n.d.-a).

2.3 Python

The programming language used when implementing the custom scripts of this experimental setup is Python. The language is widely supported, and several packages and libraries are written in the language. The libraries included in the setup are sys, pySerial, struct, and pylsl.

2.3.1 pySerial

The pySerial package is used to connect to the Shimmer devices, through a serial port.

2.3.2 Struct

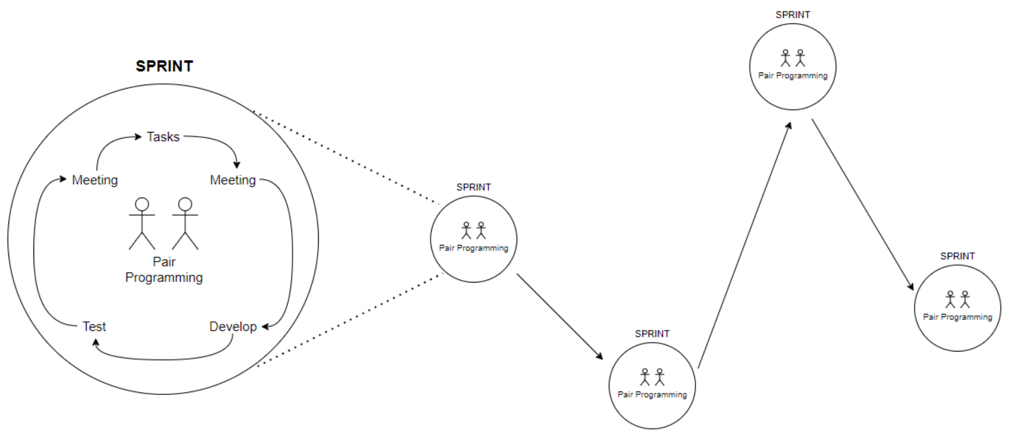
Struct is a library converting C/C++ structs to Python values. Due to the configurations made in CCS, the C structs captured by the Shimer units must be converted to Python byte objects to read the incoming data. Struct uses Format Strings, which specifies the expected layout when packing and unpacking data. The Format String consists of Format Characters and “special characters”, which specify the type of data being packed/unpacked and the byte order, size, and alignment of the data. The data types retrieved from the Shimmer3 units must correspond to the Format Characters to get the right output.

3 Agile development methodology

Three different development methodologies have been used throughout the development process. The multidisciplinary aspect of the case requires knowledge from several fields, including engineering, computer science, signal processing, and physiology. Wayfaring (Dybvik et al., 2021; Steinert & Leifer, 2012) is used as a development method within engineering design, and Extreme Programming (XP) (Sommerville, 2011) within software development, and Scrum (Gonçalves, 2018; Schwaber, 1997) as a development management method. Together, the development process makes use of the multidisciplinary environment and is an excellent example of how software development crosses with product development in the context of engineering design.

Embracing an agile development methodology was necessary to handle unknown variables, continuously new requirements, and serendipitous events, which are inevitable within product development. The Wayfaring method suggests an open approach to engineering problems and presents a journey of probing ideas. The unknown unknowns are in Wayfaring handled in Probes, where each probe is a prototype where new ideas are developed and tested. The equivalent process in XP is called a Cycle, where small frequent releases and continuous integration are encouraged by following a process consisting of meetings, development, and testing. XP consists of multiple principles, which ensures that the requirements are met. These practices include pair programming, and continuous testing and integration of code, all of which we employed in our development process. The different approaches and principles are managed by Scrum, where the focus is on integrating the complex environment of the case. The Scrum framework is made up of Sprints, which are development iterations similar to the Probe in Wayfaring. An illustration of how all methodologies are incorporated is provided in "Figure 2 - An illustration of how all development methodologies are incorporated into the process".

Figure 2 – An illustration of how all development methodologies are incorporated into the process. Inspired by the illustration of the Wayfaring process (Gerstenberg et al., 2015; Steinert & Leifer, 2012)



4 Software architecture

The agile development of the implementation of the experiment setup resulted in multiple prototypes, where each Probe/Sprint increased the understanding of the Shimmer sensors in addition to improving the previous result. The development process provided a better understanding of the Shimmer units and how to integrate the proprietary hardware into the LSL network. The result is a functioning API for multimodal sensor experiments, utilizing the LSL distribution.

All hardware must output a data stream that is recognized by the LSL network to take advantage of the open-source distribution LSL. The software architecture of the development process is provided to show how the data is captured from the Shimmer units and transmitted to the LSL network.

4.1 Shimmer software architecture

PySerial and Struct must be used in a specific manner to collect appropriate data since the Shimmer devices are pre-programmed with the firmware LogAndStream, written in C.
The architecture of the code is modular, which facilitates further development and implementation. The key features of the scripts include how pySerial and Struct functions are combined to collect the appropriate data and are as follows:

- When running the scripts, the comport where the device is connected must be included.
- It is important that the pySerial function read() and struct function unpack() is used together. Read() collects the requested data from the sensor and unpack() converts the data from C and presents the data in Python.
- The pySerial function write() and struct function pack() must be used together. Write() sends a request to the sensor, which must be converted to C using the pack() function.
- For every call to the sensor, an acknowledgement response must be called with the wait_for_ack() function. The function ensures that the request is processed correctly by the sensor and that it is ready to proceed.

4.2 LSL software architecture

To send data from Shimmer to LSL, a new stream info object must be created. The necessary parameters which must be defined include name, content type, channel count, nominal sampling rate, channel format, and source id. The following is the correct sequence in which to implement the code for creating an LSL stream from a serial port connection:

1. Open a serial port.
2. Create an LSL outlet, by defining the core stream information.
3. Read the data from the serial port.
4. Parse the signal accordingly.
5. Push the signal into the LSL outlet.

5 Open-source contribution and discussion

By utilizing LSL and enabling proprietary hardware to stream data to open-source software distributions, the development process demonstrates how to create a low-cost, multimodal physiological sensor experiment setup. New research opportunities are offered by switching from proprietary software. Higher quality, greater reliability, greater flexibility, lower cost, and the end of predatory vendor lock-in are among the promises made by open source (Open Source Initiative, n.d.-a). Utilizing LSL broadens the researcher's options by removing the financial constraint of leasing proprietary software. LSL provides several tools and resources on how to incorporate various sensors and customize the software to your need. The platform is already compatible with a wide range of sensor software, allowing data streams from numerous sensors to be synchronized. It is widely used in research and has an established community.

5.1 Advantages and drawbacks of open source

There are significant benefits to sharing research and source code. First and foremost, it promotes low-cost collaboration and innovation. Transparency and open access to information and materials allow researchers to build on each other's discoveries and ideas, which facilitates the acquisition of new knowledge. Thus, transparency promotes faster development and improvements. Another advantage is the community connected to the different projects. Collaborations, derivations, and discussions all contribute in unexpected ways to improving others' work. Approaching “the open source way” encourages everyone to accept failure as a means of improvement, and expects everyone else to do the same (Opensource.com, 2022). Forums, blog posts, papers, and journals all contribute to the ongoing growth and maintenance of projects.

However, if a project does not acquire sufficient support and has difficulty building a community, it will quickly fail. One of the principles of “the open-source way” is the idea of meritocracy. Diverse perspectives are essential to identifying the best ideas, and the most successful projects are those that receive community support and effort, even if they are not reached by consensus (Karsten, 2020). A project's survival is based on the motivation and interest in maintaining the project. In the absence of a community, or if the original contributor no longer maintains the project, it will become obsolete, if not deprecated.

When creating proprietary software or conducting research, the creation of a community is not as important as it is for open-source developers. One of the most obvious benefits of proprietary software is that it is maintained and improved by employees who are completely committed to the project. Subscription fees contribute to covering the costs of maintaining the project. Some companies also utilize the advantage to disclose some of their research or software while keeping some of the functionality confidential. In that way, they are still in control of the development in addition to keeping the economic benefits (Vanhaverbeke et al., 2008).

5.2 Adapting proprietary hardware to open-source software

Our development process evoked several questions worth discussing. Are there nuances as to what is considered open source? How many resources should a researcher use to reproduce and contribute to further development? Can Shimmer be considered open source? It is important to clarify that this is not meant as a critique against Shimmer, but rather as an evaluation of how companies take advantage of open-source properties without fully committing to open research.

Understanding the firmware of the Shimmer sensors took a lot of time and effort. Despite offering a large amount of documentation and user manuals to the customer, the information given is difficult to understand and interpret. In addition, the documentation offered by Shimmer is poorly constructed. We had to search through several user manuals and source code scripts to find associated information to fill the information gap. Shimmer makes configuring the various sensor units extremely difficult without their Consensys software tool. They do, however, claim to be "committed to transparency in explaining how our systems work, what they can and cannot do, and why, and providing and actively maintaining a huge body of product documentation", but simultaneously clarify that their support does not include detailed explanations of the engineering principles behind their software and hardware (Shimmer, n.d.).

Researchers need to know that their sensory data is correct, and Shimmer being selective in their use of transparency is an issue. The firmware settings are crucial to know as it is the foundation of what output is presented. Open-source software can be considered to be more secure and stable than proprietary software. Given the fact that open-source software can be viewed and modified by anyone, allows anyone to detect and correct errors or omissions that were missed by the original authors (Heron et al., 2013). The Shimmer units consist of hardware developed by Texas Instruments, which again is configured through their integrated development environment. It is difficult to determine what the original raw data are because various firmware handles data differently, and the corresponding software program used to show the data takes liberties and alters the output to present it in an orderly manner. There are many steps along the way where information on how to manipulate the raw data can be hidden. Due to the binary format of the firmware source code and the lack of documentation explaining the configuration, it is complicated to figure out the settings. The sensor data's integrity determines the credibility. Quality assurance provided by the firm is one of the key arguments for keeping firmware and software products proprietary. However, any solution may turn out to be just as reassuring as any proprietary solution with proper documentation and accessible source code.

Companies, such as Shimmer, benefit from open research by providing numerous scripts and documentation to any end-user by encouraging technological development without compensating for it. The advantage of the strategy is that companies, such as Shimmer, learn early on about new technologies. As a result, companies can scan a broader range of technologies or new market developments, rather than writing options on internal projects alone (Vanhaverbeke et al., 2008). They retain control and economic benefits by keeping the firmware settings to themselves. The business model appears to supply enough content to encourage the customer's interest in contributing to further development before they reach the level of confidentiality.

The answer to whether Shimmer is open source or not, there is not a definitive conclusion. However, they do embrace open innovation by publishing large amounts of documentation and harvesting the benefits that come with it. The development process uncover that the sensor units were decodable and configurable. Given the vast amount of documentation, we are confident that the data acquired are accurate data. However, one question remains unanswered: to reproduce and contribute to further development, how many resources should a researcher use? On one side, the public has access to a lot of useful information about how to handle and interpret the units' output. The question is whether the resources available, such as time and knowledge, are sufficient. On the other side, one could argue that the process is only limited to oneself. The information is available, but do you have the time and knowledge to comprehend and understand it? During the development process, the process of collecting information was time-consuming. The documentation was available, and anyone could decode and interpret the data eventually. Consequently, if you are motivated enough, it is only a matter of willpower to persevere until all questions are addressed. However, if this is the mindset to reproduce and contribute to development, all resources will run out. We assert that the relationship between resources invested, and the outcome of the final result should be fairly balanced. Within the development process described in this article, this was not.

6 Conclusion

This article adds to the field of open research by broadening the possibilities for experimental setups facilitate the acquisition of new knowledge. With the primary goal of demonstrating the benefits of open-source and how it may promote open research, we have shown how open platforms, such as LSL, may be used to save costs by eliminating the cost of leasing proprietary software.

The development case opens more research opportunities and contributes to the open research field. The accessibility provided by the case should encourage researchers to take advantage of the open-source distribution of LSL. Given that all implementation from the development process is open-source, other researchers can build upon and modify it, creating the opportunity to customize it to other purposes. Even though there is a lack of incentives to be transparent (Nosek et al., 2015), it contributes to more reliable research, and better quality, due to peer review.

We believe that physiological sensors can assist engineers in improving the design and structure of systems and products to enable more intuitive interaction with end-users, resulting in better products and, implicitly, improved productivity. This topic of study is still in its early stages. The research community can benefit from open source in the same way that software development has. Several sensors currently come with a software solution that is tailored to the sensor in question. Because the firmware is often costly, only the wealthy can conduct research. The loss of potential research due to easily accessible resources is far too great, and the area could benefit from all contributions (Pinti et al., 2020). As a result, the economic factor is dividing the field.

Code availability

The code retrieving GSR and ECG data and sending it to the LSL network is publicly available: https://github.com/catthiba/Multimodal-sensor-setup

References

Balters, S., & Steinert, M. (2017). Capturing emotion reactivity through physiology measurement as a foundation for affective engineering in engineering design science and engineering practices. *Journal of Intelligent Manufacturing*, *28*(7), 1585–1607. https://doi.org/10.1007/s10845-015-1145-2

Blessing, L. T. M., & Chakrabarti, A. (Eds.). (2009). DRM: A Design Reseach Methodology. In *DRM, a Design Research Methodology* (pp. 13–42). Springer. https://doi.org/10.1007/978-1-84882-587-1_2

Dybvik, H., Kuster Erichsen, C., & Steinert, M. (2021). Description of a Wearable Electroencephalography + Functional Near-Infrared Spectroscopy (EEG+FNIRS) for In-Situ Experiments on Design Cognition. *Proceedings of the Design Society*, *1*, 943–952. https://doi.org/10.1017/pds.2021.94

Extensible Data Format (XDF). (2021). [Python]. Swartz Center for Computational Neuroscience. https://github.com/sccn/xdf (Original work published 2015)

Gerstenberg, A., Sjöman, H., Reime, T., Abrahamsson, P., & Steinert, M. (2015). A Simultaneous, Multidisciplinary Development and Design Journey – Reflections on Prototyping. In K. Chorianopoulos, M. Divitini, J. Baalsrud Hauge, L. Jaccheri, & R. Malaka (Eds.), *Entertainment Computing—ICEC 2015* (Vol. 9353, pp. 409–416). Springer International Publishing. https://doi.org/10.1007/978-3-319-24589-8_33

Gonçalves, L. (2018). Scrum. *Controlling & Management Review*, *62*(4), 40–42. https://doi.org/10.1007/s12176-018-0020-3

Heron, M., Hanson, V. L., & Ricketts, I. (2013). Open source and accessibility: Advantages and limitations. *Journal of Interaction Science, 1*(1), 2. https://doi.org/10.1186/2194-0827-1-2

Karsten, W. (2020, December 16). *The open source way 2.0—Presenting the Open Source way*. Opensource.Com. https://www.theopensourceway.org/the_open_source_way-guidebook-2.0.html

Kothe, C., Medine, D., Boulay, C., Grivich, M., & Stenner, T. (2019). *Quick Start—Labstreaminglayer*. https://labstreaminglayer.readthedocs.io/info/getting_started.html

Nosek, B. A., Alter, G., Banks, G. C., Borsboom, D., Bowman, S. D., Breckler, S. J., Buck, S., Chambers, C. D., Chin, G., Christensen, G., Contestabile, M., Dafoe, A., Eich, E., Freese, J., Glennerster, R., Goroff, D., Green, D. P., Hesse, B., Humphreys, M., … Yarkoni, T. (2015). Promoting an open research culture. *Science, 348*(6242), 1422–1425. https://doi.org/10.1126/science.aab2374

Open Source Initiative. (n.d.-a). *About the Open Source Initiative*. Retrieved 15 February 2022, from https://opensource.org/about

Open Source Initiative. (n.d.-b). *The Open Source Definition*. Retrieved 1 December 2021, from https://opensource.org/docs/definition.php

Opensource.com. (2022). *What is open source?* Opensource.Com. https://opensource.com/resources/what-open-source

Pinti, P., Tachtsidis, I., Hamilton, A., Hirsch, J., Aichelburg, C., Gilbert, S., & Burgess, P. W. (2020). The present and future use of functional near-infrared spectroscopy (fNIRS) for cognitive neuroscience. *Annals of the New York Academy of Sciences*, *1464*(1), 5–29. https://doi.org/10.1111/nyas.13948

Schwaber, K. (1997). SCRUM Development Process. In J. Sutherland, C. Casanave, J. Miller, P. Patel, & G. Hollowell (Eds.), *Business Object Design and Implementation* (pp. 117–134). Springer London. https://doi.org/10.1007/978-1-4471-0947-1_11

Shimmer. (n.d.). Contact Support. *Shimmer Wearable Sensor Technology*. Retrieved 1 December 2021, from https://shimmersensing.com/support/wearable-sensing-support/

Shimmer Research. (n.d.-a). *GSR User Guide rev 1.13*.

Shimmer Research. (n.d.-b). *Shimmer Wearable Sensor Technology*. Shimmer Wearable Sensor Technology. Retrieved 15 February 2022, from https://shimmersensing.com/

Shimmer Research. (2017). *Shimmer User Manual*.

Shimmer Research. (2018). *LogAndStream for Shimmer3 Firmware User Manual*.

Sommerville, I. (2011). *Software engineering* (9th ed). Pearson.

Steinert, M., & Leifer, L. J. (2012). 'Finding One's Way': Re-Discovering a Hunter-Gatherer Model based on Wayfaring. *International Journal of Engineering Education*, *28*(2), 251.

Texas Instruments. (n.d.-a). *ADS1292R data sheet, product information and support | TI.com*. Retrieved 14 February 2022, from https://www.ti.com/product/ADS1292R

Texas Instruments. (n.d.-b). *CCSTUDIO-C2000 IDE, configuration, compiler or debugger*. Retrieved 21 November 2021, from https://www.ti.com/tool/CCSTUDIO-C2000

Texas Instruments. (n.d.-c). *MSP430F5437A data sheet, product information and support*. Retrieved 21 November 2021, from https://www.ti.com/product/MSP430F5437A#features

Vanhaverbeke, W., Van de Vrande, V., & Chesbrough, H. (2008). Understanding the Advantages of Open Innovation Practices in Corporate Venturing in Terms of Real Options. *Creativity and Innovation Management, 17*(4), 251–258. https://doi.org/10.1111/j.1467-8691.2008.00499.x

Chapter 5

Pilot Experiment

The pilot experiment gives an example of how to conduct a multimodal physiological sensory experiment using the LSL distribution. The sensors used in the setup are ECG, GSR, EEG, fNIRS, and eye-tracking. The experiment is a questionnaire created using PsychoPy. By connecting the sensors and PsychoPy to LabRecorder, every response captured by the sensors is synchronized to the stimuli onset.

5.1 Experiment setup

As said earlier, the experiment setup presented consists of multiple physiology sensors, connected to the same software recording program, which are EEG, fNIRS, ECG, GSR, and eye-tracking. Besides the use of different sensors than the original study executed with functional magnetic resonance imaging (fMRI), it follows the same experiment setup.

The experiment is designed using the PsychoPy software program, mainly implemented by Abelson (2021). Compared to the original study there are some overall changes regarding how the experiment was executed. Due to the constraining conditions of an MRI scanner, the participant had to lay down and view the stimuli on a monitor through a look-out mirror attached to a head-mounted coil and use responsive gloves strapped to their hand to indicate new ideas and give questionnaire ratings. The current experiment setup is much less invasive, allowing the participant to sit in a chair in front of a monitor equipped with a standard computer mouse and keyboard to indicate new ideas and submit questionnaire ratings.

A table of the technical information regarding each sensor is listed in Table 5.1.

5.2 Experiment procedure

The problems and inspirational stimuli used in this experiment are identical to those used in the original study by Goucher-Lambert et al. (2019). The inspirational stimulus is a subset of the

Hardware	Manufacturer		Information
EEG	OpenBCI		f_{sample}: 250 Hz
fNIRS	NIRx		f_{sample}: 7.81 Hz
GSR	Shimmer		f_{sample}: 50 Hz
ECG	Shimmer		f_{sample}: 512 Hz
Eye tracker	Pupil Core	World cam.	Resolution: 1280x720 pixels f_{sample}: 30 Hz Field of view: 99 degrees x 53 degrees
		Eye cam.	Resolution: 192x192 pixels f_{sample}: 120 Hz

Table 5.1: Hardware specifications

extracted words gathered through the prior research study from Goucher-Lambert and Cagan (2017), where they were obtained by combining crowd-sourcing and text-mining techniques. The problems are modified versions of problems used by the design-by-analogy literature. Table 5.2 lists the exact problems, including the citation of the original source, and inspirational stimuli used in the experiment.

Problem	Near Words	Far Words	Control Words
1. A lightweight exercise device that can be used while traveling (Linsey & Viswanathan, 2014)	pull, push, band, resist, bar	roll, tie, sphere, exert, convert	lightweight, exercise, device, while, traveling
2. A device that can collect energy from human motion (Fu et al., 2013)	store, charge, shoe, pedal, step	beam, shake, attach, electrons, compress	device, collect, energy, human, motion
3. A new way to measure the passage of time (Tseng et al., 2008)	light, sand, count, fill, decay	crystal, drip, pour, radioactive, gravity	new, way, measure, passage, time
4. A device that disperses a light coating of a powdered substance over a surface (Linsey et al., 2008)	spray, blow, fan, shake, squeeze	rotor, wave, cone, pressure, atomizer	light, coating, surface, powdered, substance
5. A device that allows people to get a book that is out of reach (Cardoso & Badke-Schaub, 2011)	extend, clamp, pole, hook, reel	pulley, hover, sticky, voice, angle	device, allows, people, book, reach
6. An innovative product to froth milk (Toh & Miller, 2014)	spin, whisk, heat, shake chemical	surface, pulse, gas, gasket, churn	an, innovative, product, froth, milk

7. A way to minimize accidents from people walking and texting on a cell phone (Miller et al., 2014)	alert, flash, camera, sensor, motion	emit, react, engage, lens, reflection	minimize, accidents, walking, texting, phone
8. A device to fold washcloths, hand towels, and small bath towels (Linsey et al., 2012)	robot, press, stack, table, rotate	deposit, cycle, rod, funnel, drain	fold, wash, cloths, hand, towels
9. A way to make drinking fountains accessible for all people (Goldschmidt & Smolkov, 2006)	adjust, lift, hose, hose, nozzle	shrink, catch, attach hydraulic, telescopic	way, drinking, fountains, accessible, people
10. A measuring cup for the blind (Jansson & Smith, 1991; Purcell et al., 1993)	braille, touch, beep, sound, sensor	preprogram, recognize, pressure, holes, cover	measuring, cup, for, the, blind
11. A device to immobilize a human joint (Wilson et al., 2010)	clamp, lock, cast, harden, apply	shrink, inhale, fabric, condense, pressure	device, to, immobilize, human, joint
12. A device to remove the shell from a peanut in areas with no electricity (Viswanathan & Linsey, 2013)	crack, crank, blade, squeeze, conveyor	melt, circular, wedge, chute, wrap	device, remove, shell, peanut, areas

Table 5.2: Problem statements and inspirational stimuli used for the experiment

The task of the experiment was a conceptual design-thinking task, where participants were asked to develop as many ideas as possible for 12 open-ended design problems, within the allotted time for each. An idea was indicated by pressing space on the keyboard so that the human response at that time could be examined. In the experiment, there were three different conditions: two with inspirational stimuli, either near or far, and a third with control words - words from the design problem. The participants were broken into three counterbalanced groups, where each participant was presented with 4 problems from each condition. The conditions given to the different groups are listed in Table 5.3.

Problem	1	2	3	4	5	6	7	8	9	10	11	12
Group A	Near	Cntrl	Far	Near	Far	Cntrl	Far	Cntrl	Near	Cntrl	Far	Near
Group B	Far	Near	Cntrl	Far	Cntrl	Near	Cntrl	Near	Far	Near	Cntrl	Far
Group C	Cntrl	Far	Near	Cntrl	Near	Far	Near	Far	Cntrl	Far	Near	Cntrl

Table 5.3: Condition by the problem for each experimental group

A visualization of the experiment outline is presented in Figure 5.1 , with timings for each problem and an example of the stimuli presentation. The experiment starts with a self-paced instruction screen, allowing the participant to start when ready. The experiment follows the same outline for each problem, starting with a presentation of the design problem displayed for 7 seconds. Following is a fixation cross routine, indicated by ”+”, breaking up the viewing of the design problem and the start of stimuli presentation. The fixation cross routine allows distinguishing between brain activity connected with the initial problem presentation and brain activity associated with inspired stimuli (Goucher-Lambert et al., 2019). Participants were given two minutes to come up with design solutions. The two minutes were divided into two blocks of one minute each, separated by an addition task named 1-Back memory task. The first block included *Word Set 1*, where only three out of the five inspirational stimuli were provided. In that way, the presentation of inspirational stimuli was spread throughout the problem-solving period. The second problem-solving block, *Word Set 2*, included the remaining two words. The additional stimuli provide a mechanism for new connections if the participant had exhausted their use of the inspirational stimuli in *Word Set 1* (Goucher-Lambert et al., 2019).

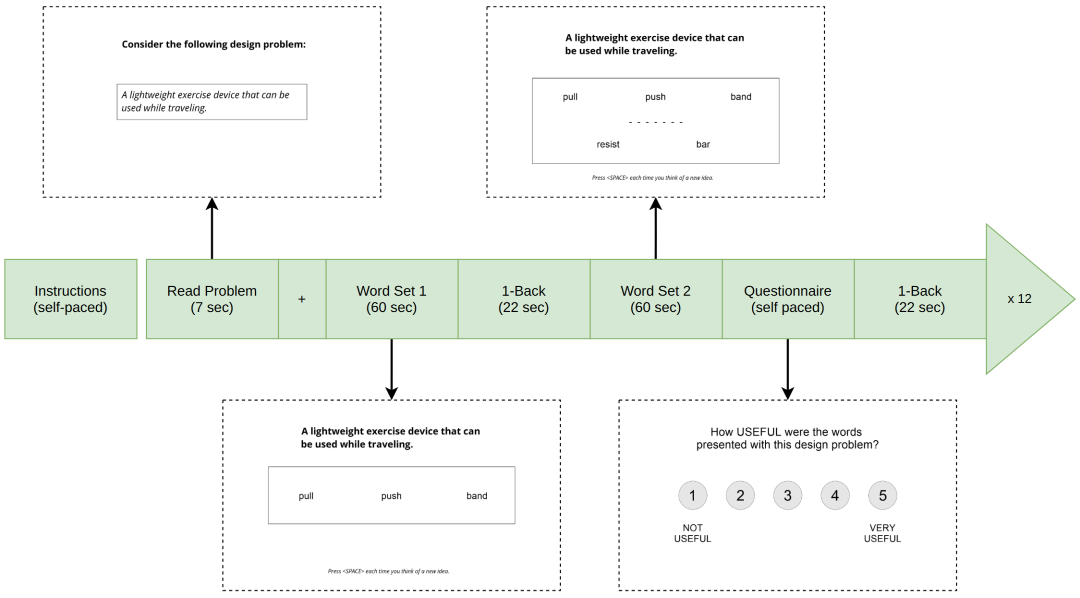

Consider the following design problem:
A lightweight exercise device that can be used while traveling.
A lightweight exercise device that can be used while traveling.
pull
push
band
resist
bar
Press <SPACE> each time you think of a new idea.
Instructions (self-paced)
Read Problem (7 sec)
+
Word Set 1 (60 sec)
1-Back (22 sec)
Word Set 2 (60 sec)
Questionnaire (self paced)
1-Back (22 sec)
x 12
A lightweight exercise device that can be used while traveling.
pull
push
band
Press <SPACE> each time you think of a new idea.
How USEFUL were the words presented with this design problem?
1
2
3
4
5
NOT USEFUL
VERY USEFUL

Figure 5.1: Problem outline with timing and stimuli example

The 1-Back memory task involves the participant indicating whether or not a single letter displayed on the screen matched the previous letter. The task was included in between the two blocks of inspirational stimuli, in order to allow for the hemodynamic response related to design ideation to return to the base level.

After *Word Set 2* the participant was asked through a questionnaire to rate the inspirational stimuli and their subjective perception regarding their solutions. The four questions can be viewed below, and were all rated from 1 - ”not useful” to 5 - ”very useful”.

1. How USEFUL were the words presented with this design problem?
2. How RELEVANT were the words presented with this design problem?

3. How was the overall NOVELTY (uniqueness) of the solutions you developed for this design problem?

4. How was the overall QUALITY of the solutions you developed for this design problem?

5.3 Experiment implementation

The implementation of the experiment setup is based on previous experiment setups developed by TrollLABS, however, new configurations are implemented to allow LSL connection. It combines EEG and fNIRS (Dybvik, Kuster Erichsen, et al., 2021), eye tracking technology (Abelson, 2021), ideation experiment (Goucher-Lambert et al., 2019), and ECG and GSR into one single experiment setup, utilizing the open-source distribution LSL (Kothe et al., 2019b) and PsychoPy (Peirce et al., 2019).

Both EEG and fNIRS is integrated into the same cap, as described by Dybvik, Kuster Erichsen, et al. (2021). A special adapter was developed in order to mount the spring-loaded EEG from OpenBCI (Erichsen et al., 2020) onto the already functioning fNIRS system, in addition to the Cyton board and battery. The complete setup of the cap enables low-cost integration of EEG with fNIRS for multimodal brain imaging (Dybvik, Kuster Erichsen, et al., 2021). The EEG is connected to the computer through a dongle, making the connection wireless. The EEG data stream is converted to an LSL stream through a custom Python script, see Appendix D. The fNIRS is connected to the laptop directly. Using NIRStar 15.2, the fNIRS data stream is forwarded to LSL.

Pupil Labs' eye-tracker has a *Surface Tracker* plugin, allowing planar surfaces to be defined and tracked. The surface is defined with small binary markers called *April Markers.* Once the surface definitions are added to the directory they can be reused and accessed for each session. The eye tracker is directly connected to the laptop using a USB wire. Pupil Capture enables LSL streaming sending the gaze data to LabRecorder.

The implementation of the experiment is mainly programmed by Abelson (2021), using the open-source software PsychoPy v2021.1.4, however, some alterations are made to include all necessary information from PsychoPy to the stream sent to LSL. Most of the experiment's input data and the visual design originate from figures and tables from the original study (Goucher-Lambert et al., 2019). The problem statements and inspirational stimuli were loaded into different Excel files, according to the different condition groups. PsychoPy creates its own LSL stream sending the annotations and timestamps to LabRecorder.

Both the ECG and GSR sensors are connected to the laptop through Bluetooth. The data stream is converted to an LSL stream through the custom Python scripts shown in Appendix B, and Appendix C. Both EEG, ECG, and GSR run concurrently by running the *main.py* scrip, Appendix A.

5.3.1 Data quality and calibration

In order to obtain valid data from which to draw conclusions, researchers must follow a consistent application protocol for applying sensors. Given the importance of signal quality, a description of how to obtain high-quality data is provided.

Both the eye tracker and fNIRS system can cause electrical inference and distort the EEG signals. Hence the wire configuration on the cap and placement of the eye-tracking frame must be addressed to minimize the distortion. The distance between the leads should be maximized. When placing the cap, both the optodes and the electrodes require direct scalp contact. The quality of the signal can be affected by hair thickness, the ease with which the hair can be parted, as well as the conductivity of the skin.

Before recording, both EEG and fNIRS require a visual evaluation of signal quality. The EEG signal is evaluated in OpenBCI, which displays a live time-series plot and an FFT plot indicating the frequency distribution for each channel. Crosstalk with either fNIRS or the eye tracker will be indicated by peaks in the FFT plot at harmonics of the given sampling rate to the corresponding sensor. Calibration of the fNIRS signal is required before it can be evaluated. The *Quality Scale Tool* in NIRStar illustrates the integrity of the incoming data by color: green, yellow, and red. All channels should be green, however yellow can be accepted.

Calibration of the eye tracker must also be performed, in order to establish a mapping between pupil and gaze position. A validation routine should also be performed, to ensure the accuracy error is not too high (Labs, 2022).

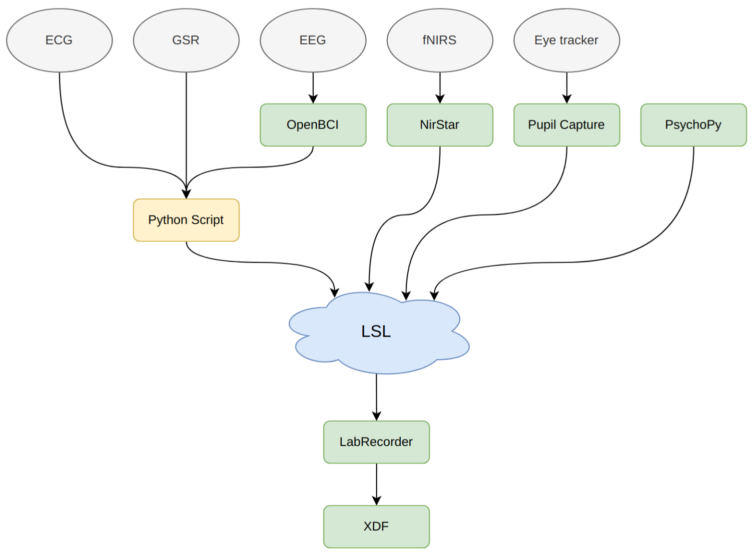


Figure 5.2: A flow chart illustrating how the various sensors (gray), software (green), and custom Python scripts (yellow) are connected to the lab streaming layer and recorded by LabRecorder, which compiles the data into a single XDF file.

5.4 Running the pilot experiment

To ensure a fully functional experiment setup, several pilot experiments were conducted to make sure that everything is performed as planned. Given the thorough work by Abelson (2021) when implementing the experiment in PsychoPy, all technical bugs and unclear task descriptions were resolved. The main focus of the pilot experiment is therefore to ensure all sensors are compatible and that the laptop can withstand the workload of the data gathering. The experiment setup was therefore only used for a fraction of the total problem statements to provide quick feedback. Only the first three problem statements and inspirational stimuli from condition group A were used before a full run-through of all twelve questions was executed.

The sensors were placed on the participant in the following order. First, the ECG was placed on the upper body, as described in Section 6.3. Second, the fNIRS and EEG cap were placed on the participant's head. The eye tracker is placed over the cap, in order to prevent the frame of the glasses to lift the cap from the participant's head. Lastly, the GSR is placed on the non-dominant hand of the participant.

The experiment protocol

The setup requires multiple software programs to run in concurrency, as it is visualized in Figure 5.2. LabRecroder is the recording program, in which all the data streams are collected. Following is a list of how to start sending the data streams.

1. Open LabRecorder
2. Open the advanced Bluetooth settings and ensure which comports belong to which Shimmer sensor.
3. Open OpenBCI and ensure all electrodes are receiving signal. The program must be closed after configuration, in order for the script to connect to the device.
4. Open NIRStar
   (a) Run the configuration
   (b) Run the preview of data, this will automatically create an LSL stream
5. Open Pupil Capture and follow the recommended protocol according to Pupil Labs (Labs, 2022)
   (a) Start recording in Pupil Capture.
   (b) Ask the participant to slowly look around while keeping the head stable for a couple of seconds to sample different gaze angles. This will allow the eye model to stabilize at the start.
   (c) Perform a calibration.
   (d) Perform a validation. If the error is too high, restart the block.
   (e) Perform your experiment.

6. Open a terminal and navigate to the project folder.
   (a) Run the *main.py* file with two arguments, the first being the GSR comport and the second the EEG comport. Now both ECG, GSR, and EEG are streaming data to LSL
7. Open PsychoPy and start the experiment
8. Update LabRecorder and select all streams. Start Recording.
9. Perform the experiment
10. Stop recording in LabRecorder after the experiment has ended
11. Stop the streams from all sensors
    (a) End script
    (b) Stop preview in NirStar Stop recording in Pupil Capture
12. An XDF file is created and can be used as needed.

Chapter 6

Data Processing

With a multimodal setup, it is important to know how to interpret the data gathered from the various sensors. Establishing a valid data set is challenging, and there are numerous factors that must be considered when creating the analysis protocol. Especially in situ experiments are prone to invalid data gathering. When analyzing the data, it is important to take into account how the challenges contributed to the outcome.

Examples of common sources of error include placement of sensors, sampling rate, external stimuli, and noise sensitivity (Balters & Steinert, 2017). As mentioned earlier, placement is crucial and has a huge impact on the data output. The sampling rate affects how the response is captured, which is handled differently by each sensor. There has to be a balance between accuracy and the amount of information gathered. Regardless of whether the experiment is conducted within laboratory settings or in situ, the external stimuli have a major impact on the measurements and must be taken into account. Considering the ideation experiment discussed in this thesis, the influence of external stimuli could be adjusted before conducting the experiment. An experiment conducted in situ, however, must be adjusted for external factors during data analysis. The electrical measurement tools described in this thesis, ECG and EEG, are inherently noise sensitive. Errors may be caused by the improper attachment of electrodes and by body movement.

This section gives an overview of how the sensors are placed on the participant, how they are connected to the laptop running the experiment, and how to evaluate and interpret the data measurements from the various sensors.

6.1 EEG

For the collection of EEG data, the electrodes were mounted according to the five percent system (Oostenveld & Praamstra, 2001), with the placement of channels specified in Table 6.1, and visualized in Figure 6.1 where the EEG electrodes are marked in red. The setup consists of a Cyton biosending board from OpenBCI with 8 spring-loaded, dry electrodes provided by Ultracortex, seen in Figure 6.2. A custom Python script captures the data from the Cyton board and sends the data to LabRecorder. The sampling rate is set to 250 Hz.

Channel	**Location**
1	AFp1
2	Afp2
3	C3
4	C4
5	P7
6	P8
7	O1
8	02

Table 6.1: EEG electrode placement according to the five percent system (Oostenveld & Praamstra, 2001)

EEG is an electronic measurement tool making it inherently prone to noise. The sources of error in EEG data are many and must be considered when analyzing the measurements. Due to EEG's low spatial resolution, it's difficult to determine precisely what brain areas are being activated (Antonenko et al., 2010; Balters & Steinert, 2017). Furthermore, EEGs are sensitive to motion artifacts, such as blinking and body movements, which sometimes can be stronger than neural activity. Eye movements (blinking) are a major source of contamination of EEG, as the retina causes a change in the electric fields that surrounds the eyes, which in turn distort the electric fields over the scalp (Croft & Barry, 2000). Another source of noise is other electrical devices, which in this case can both be affected by the eye tracker and fNIRS.

The interpretation of the vast amount of data from an EEG recording is difficult, especially since the brain wave patterns are unique for every individual (Abhang et al., 2016).

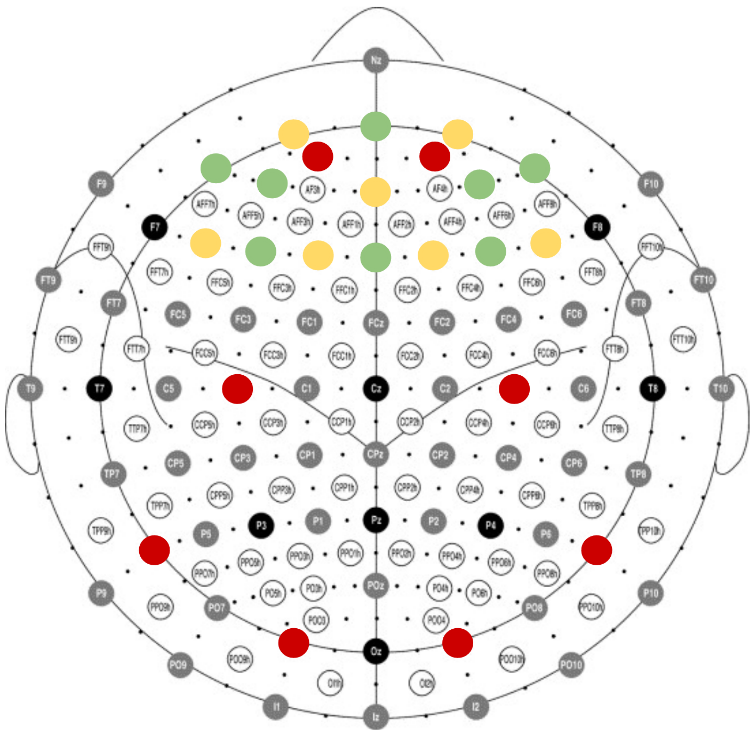


Figure 6.1: The collection of EEG and fNIRS data utilizes the setup developed by Dybvik, Kuster Erichsen, et al. (2021). The cap integrates both EEG and fNIRS. Optodes and electrodes are positioned according to the five percent system (Oostenveld & Praamstra, 2001). The EEG electrodes are marked in red, and the fNIRS optodes are marked in green (sources) and yellow (detectors).

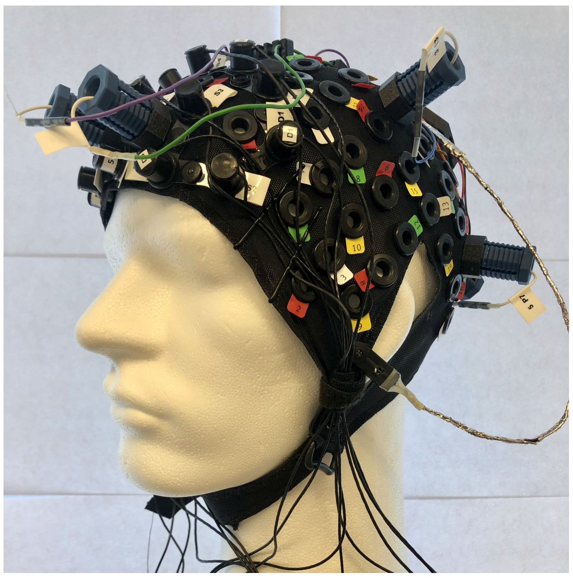


Figure 6.2: The cap with EEG electrodes and fNIRS optodes integrated

6.2 fNIRS

The fNIRS data is collected through the NIRSport system, with 8 sources and 7 detectors, listed in Table 6.2. The optodes are positioned over the prefrontal cortex, which can be viewed in Figure 6.1 and Figure 6.2. The two wavelengths (760 nm and 850 nm) are sampled at 7.81 Hz. A wire connects the fNIRS device to the laptop.

Source	**Location**	**Detector**	**Location**
S1	F3	D1	F5
S2	AF7	D2	F1
S3	AF3	D3	Fp1
S4	Fz	D4	AFz
S5	Fpz	D5	F2
S6	AF4	D6	Fp2
S7	F4	D7	F6
S8	AF8	-	-

Table 6.2: fNIRS optodes placement according to the five percent system (Oostenveld & Praamstra, 2001)

Commonly, the fNIRS measurement tool has a temporal sampling rate up to 10 Hz, allowing better tracking of the hemodynamic response than fMRI which has a lower temporal resolution. Even though the fMRI has a good tolerance for motion artifacts, it may appear as fast and narrow spikes or shifts from baseline values in the fNIRS signal (Pinti et al., 2020). Another source of error that can influence fNIRS data is how the systemic blood flow changes on hemodynamic signals caused by body movements. As a consequence, fNIRS signals can produce both false positives and false negatives with regard to the statistical inference of functional activity, since fNIRS signals are composed of both neuronal activity and systemic elements (Pinti et al., 2020).

The field of fNIRS has grown rapidly, however, it is important to highlight the fact that there is neither an agreement nor guidelines on how to analyze the fNIRS data (Hocke et al., 2018; Pinti et al., 2020). The lack of standardization and automated processing- and analyzing methods can lead to poor quality studies, or misinterpretation and replication difficulties (Hocke et al., 2018).

6.3 ECG

For collecting ECG data a Shimmer3 EXG Unit is used (Research, 2017a). The Shimmer3 EXG unit consists of five electrodes, creating four lead situations. The electrodes are placed by the left- and right arm, left- and right leg, and one by the ribs (V5), displayed in Figure 6.3. V5 allows for the highest quality capturing of R waves (Research, 2018). The leads captured by the unit are from the right arm to the left arm, the left leg to the right arm, the left leg to the left arm, and Wilson's Central Terminal (WCT), which is the mean voltage obtained from the three previous leads. The WCT represents the average potential of the body and functions as a reference point (Research, 2018). The sampling rate is set to 512 Hz. The ECG data is streamed to LSL through a custom script over a Bluetooth connection to the laptop running the experiment.

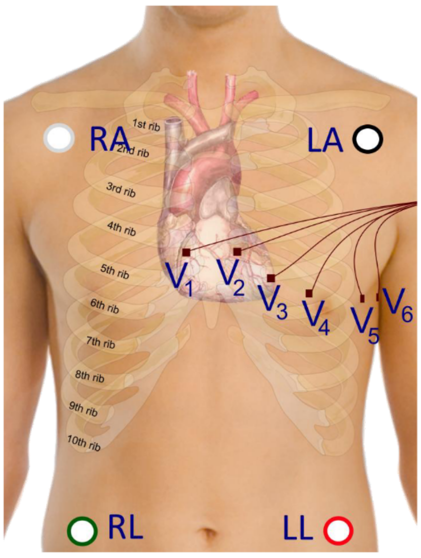


Figure 6.3: The placement of ECG electrodes, the picture is obtained from (Research, 2018)

The ECG needs a high sampling rate in order to capture the electrical activity of the heart, due to the short duration of the waves included in the ECG waveform, see Section 3.3. It can be advantageous to isolate the data measurements within the timespan of the stimuli, due to the massive amount of data captured. In addition, the duration of both the P-wave and the QRS complex is 0.12 seconds (Gacek & Pedrycz, 2011). Changes in values should be concentrated around these timestamps.

The ECG being an electrical measurement tool, is vulnerable to numerous recording errors. One of them is the drifting of the electrodes due to respiration. Activation of the sweat glands may cause electrode impedance, and body movement will cause muscle noise. Muscle noise is the worst source of error as it usually overlaps the actual ECG data. The use of low-pass filters may reduce the high-frequency components, but there is no filter that solves the problem. Another source of error is electromagnetic radiation from other electrical devices (Balters & Steinert, 2017). Ensuring nearby devices are at an appropriate distance may solve this problem.

6.4 GSR

For the collection of GSR data, a Shimmer3 GSR+ Unit is used (Research, 2017b). Palms and fingers have a high density of sweat glands, which is why the GSR electrodes are placed on the inside of the palm on the index and middle fingerFigure 6.4. The sampling rate is set to 128 Hz. The GSR data is streamed to LSL through a custom script over a Bluetooth connection to the laptop running the experiment.

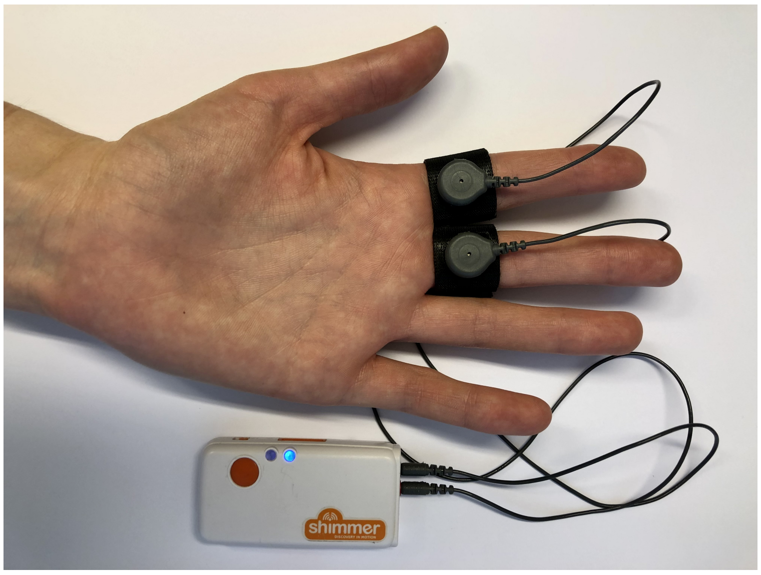


Figure 6.4: The placement of GSR electrodes

The skin conductance response (SCR) is displayed as a steep incline to the peak, and a slow decline to baseline (Benedek & Kaernbach, 2010). Only within a delay window of 0.8-4 seconds is the SCR from arousal considered "valid and significant". Lower reactions are not considered stimulus-related (Balters & Steinert, 2017). A minimum amplitude criterion (0.05 μS) is frequently used as well (Benedek & Kaernbach, 2010). The GSR should be interpreted in the context of individual differences, by finding their baseline and looking for changes in the GSR rather than analyzing absolute values.

The galvanic skin response is highly influential and varies greatly from participant to participant. The individual amount of sweat glands, dryness of skin, room temperature, direct exposure to sunlight, physical body movement, and other physiological factors may activate the sweat glands and alter the GSR conductance level (Balters & Steinert, 2017). Given all of the variables that could influence GSR data, it's best to utilize a GSR sensor under carefully controlled static physical settings, which implies it's not suited for in situ experiments.

6.5 Eye tracker

The eye-tracking data is collected by a head-mounted, binocular eye tracker from Pupil Labs (Kassner et al., 2014). The wearable eyeglasses-like frame consists of two inward-facing eye cameras, and one forward-facing world camera, which can be seen in Figure 6.5. The Pupil Lab software distribution consists of Pupil Capture and Pupil Player. Pupil Capture uses the pupil detection algorithm which uses the dark pupil effect, where the infrared light is directed away from the camera's optical axis (Kassner et al., 2014). The eye-tracker is directly connected to the laptop via a wire.

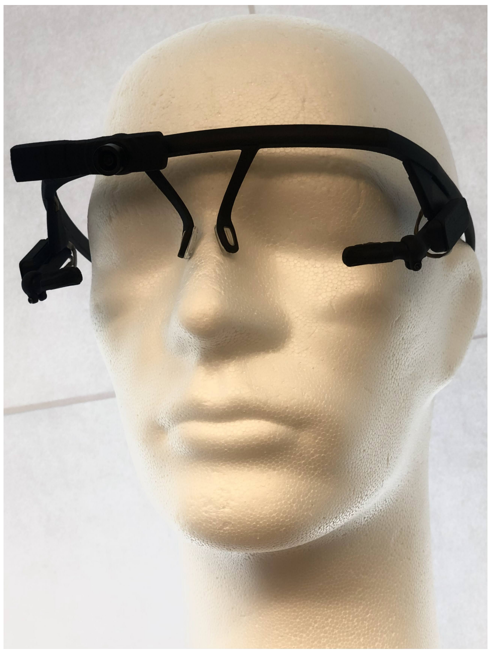


Figure 6.5: The eye-tracker from Pupil Labs

Combining eye tracking with other measurement tools will add a new dimension to how to measure the human response. As mentioned earlier, the eye-mind correlation reflects the mental processing of what the participant is looking at in the given moment, and adding it to a multimodal experiment setup will contribute massively. However, there are some factors that must be addressed when analyzing the data. The real-time data gathering of the eye tracker can provide a moment-by-moment insight into unfolding cognition. Due to eye movements generally being outside of conscious control, it can be used to tap into non-conscious processing (Carter & Luke, 2020).

The quality of eye-tracking data is defined by accuracy and precision, where the accuracy refers to the measured eye position corresponding to the actual eye position, and precision is provided by consistent measurements of the position of the eye (Carter & Luke, 2020). The primary sources of error which affect the data quality are participant characteristics, equipment features, and experimental setup (Blignaut & Wium, 2014). Participant characteristics include differences in eye shape or size, ethnicity, view angle, eye color, the position of the iris within the eye socket, and the state of the eye when it is open or closed. As the eye's darkest point is assumed to be the pupil, even the use of mascara will disturb the eye-tracking algorithm. Equipment-related problems include unstable or unsuitable sampling frequency, low camera resolution, incorrect identification of the pupil, calibration procedure, and other hardware- and software-related issues. And lastly, experimental conditions where light condition, head position, and stimulus can vary greatly are factors that can largely affect the outcome (Blignaut & Wium, 2014).

Chapter 7

Results

The main goal of this project was to have a low-cost functional experiment setup, which mainly utilizes open-source. It is not within the scope of this thesis to give a statistical analysis of the recorded data. However, a demonstration of how to perform the various analyses is given. NeuroKit2 is used to analyse and visualize features of GSR and ECG, and MNE is used to analyse EEG and fNIRS. The main reason for the demonstration is to show that the setup, in fact, can be used to perform advanced analysis comparable to the ones in related literature.

How the analysis is performed is provided in Appendix F, Appendix G, and Appendix H. The process of gathering and visualizing the results, started by creating simple plots to get a better understanding of the recorded data. The simple plot given in Figure 7.1, shows the measurements from all sensors, without ECG, synchronized to the same timestamps. The reason why ECG is left out is due to the high sample frequency, which results in a compact distribution due to the long period of time and high sampling rate. A more comprehensive analysis is then executed with the use of NeuroKit2 and MNE.

7.1 Observations

Observations of the plot reveal that the accelerometer measurements from the EEG, which are the features that measure if the participant is moving their head, are indeed very helpful when looking for artifacts in other measurements. As one can see in Figure 7.1, the oscillations with spikes differing from the baseline, indicate movement of the head. As a result, both the signal captured from EEG and fNIRS show signs of artifacts in the same time interval, most likely due to poor connection or misplacement of the EEG electrode or fNIRS optode. In addition, the movement does have a valid explanation. The participant had to move the hand from the space button to the mouse in order to answer the questionnaire. As one can confirm with the eye-tracking data, the participant did move the head to ensure the placement of the mouse. A video of the session can be found by clicking this link: **Ideation experiment**.

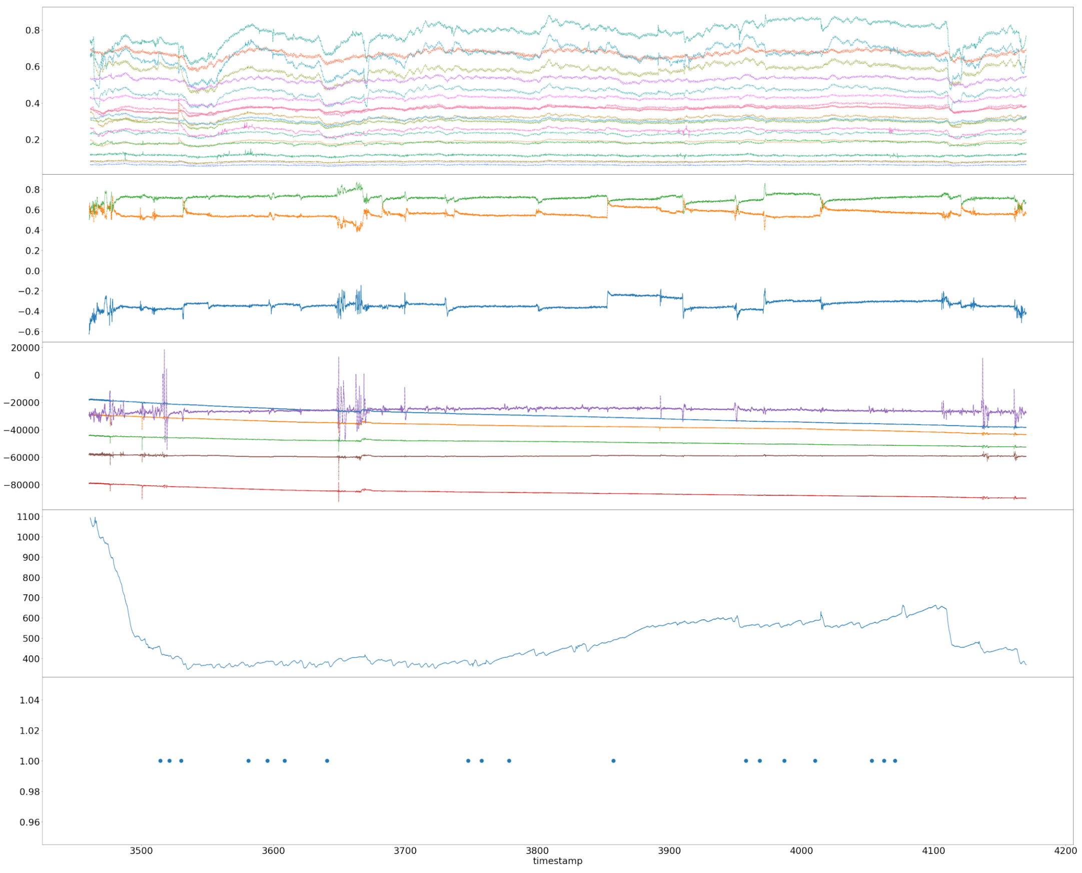


Figure 7.1: All sensors of Participant 1. fNIRS at the top, then EEG_AUX, EEG, GSR, and Psychopy - with markers from when ideas were generated

Question	Feedback			
	Q1: Usefulness	Q2: Relevancy	Q3: Uniqueness	Q4: Quality
1	1	1	1	1
2	1	1	1	1

Table 7.1: Feedback results

7.2 Analysis tools

Providing a statistical analysis and explanation of the results are too comprehensive to include in this thesis, and is not considered within the scope. However, a demonstration on how to use analysis tools and their methods is provided. Given it is only performed as a pilot experiment, there can not be drawn any conclusions from the analysis.

7.2.1 NeuroKit2

The following plots is the result from the analysis example given by NeuroKit. The execution of GSR and ECG analysis can be viewed in Appendix F.

GSR

Figure 7.2 is a plot where the SCR features are located. The features include the location of 1) peak onsets, 2) peak amplitude, and 3) half-recovery time.

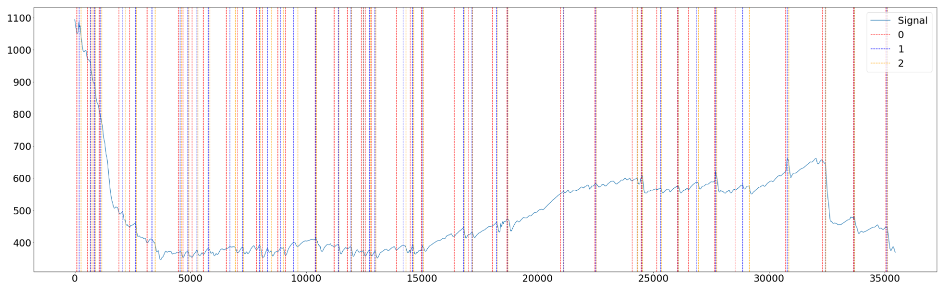


Figure 7.2: NeuroKit2 detection of SCR, where 0 is peak onsets, 1 is peak amplitude and 2 is half-recovery time.

All features extracted from the GSR data is presented in Figure 7.3. This includes raw and cleaned signal, SCR with peak onsets, peak amplitude, and half recovery time, in addition to the skin conductance level (SCL).

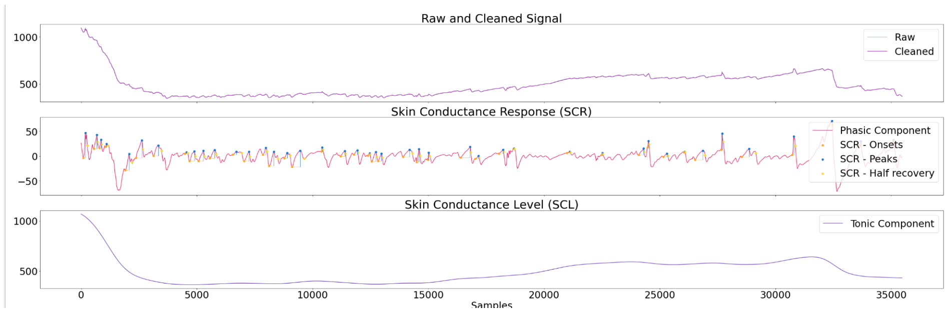


Figure 7.3: NeuroKit2 plot of all GSR features

ECG

A short time interval is chosen to perform the ECG analysis. Due to the high sampling rate it is not sufficient to include the entire dataframe. Figure 7.4 visualizes detection of R-peaks, which is the prominent feature of the QRS complex described in chapter 3.1.3. Observations from the plot reveal that the signal wave deviates from the typical heart beat waveform. However, it is worth noticing that the data still can be used to perform analysis, such as detection of R-peaks, which are detected, and segmentation of heart beats, seen in Figure 7.5.

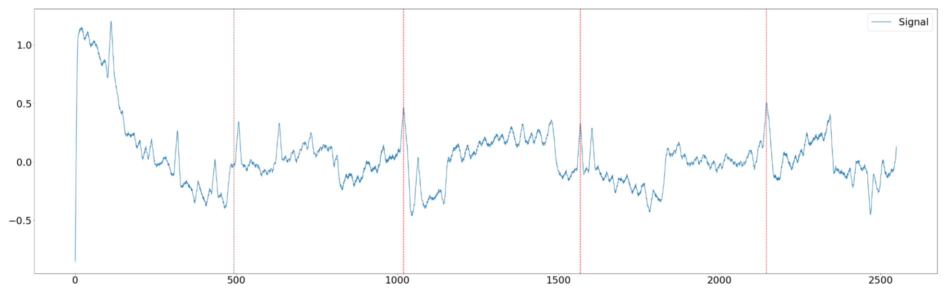


Figure 7.4: NeuroKit2 detection of R-peaks in the heartbeat

Since all R-peaks are detected in Figure 7.4, it is possible to delimit the time interval around them, and synchronize all individual heart beats, which can be seen in Figure 7.5. The irregular waveform of the ECG data is prominent when synchronizing the heart beats, however the R peak can be seen at time 0, according to the plot in Figure 7.5.

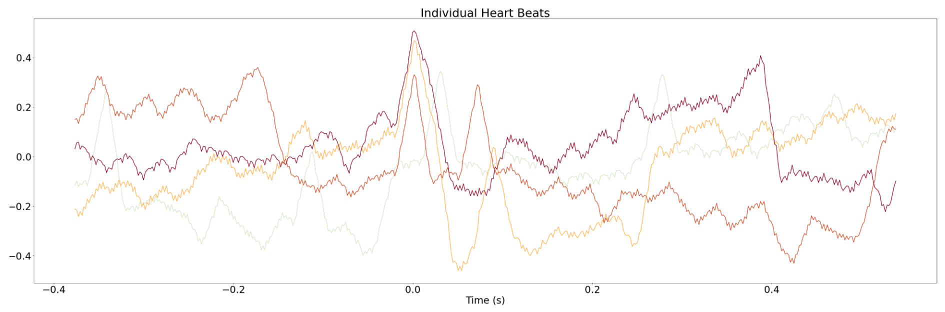


Figure 7.5: NeuroKit2 collection of all heartbeats placed on top of each other

7.2.2 MNE

The analysis of EEG and fNIRS show that the data captured through LSL can be used to perform MNE analysis. However, further preprocessing of the data must be performed in order to utilize all functions of the MNE tool. This is not within the scope of this thesis, and is therefor suggested as further work. The execution of EEG analysis can be viewed in Appendix G, and fNIRS can be viewed in Appendix H.

EEG

The MNE analysis tool is able to read XDF files, which is then converted into a raw arrays. The first half of the recorded data is displayed in Figure 7.6, which has the same outline as the EEG plot in Figure 7.1. This indicates that the conversion is done correctly.

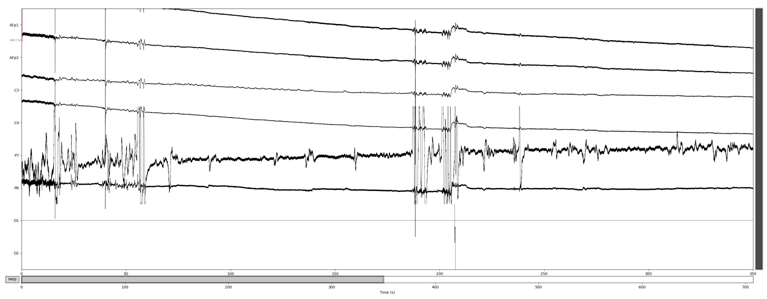
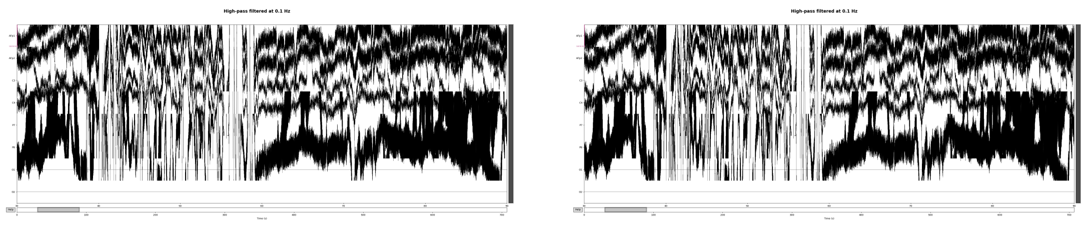


Figure 7.6: MNE plot of EEG measurements

After the data is converted into a raw array, it is possible to utilize MNE methods, such as filtering. Artifacts, such as drifting, can sometimes be repaired by filtering. Depending on half period the time interval the drift appears to last, the high-pass should be higher than that, to ensure those components are excluded. **??** shows a filter of 0.1 Hz, and **??** shows a filter of 0.5 Hz.



(a) High-pass filtered at 0.1 Hz (b) High-pass filtered at 0.5 Hz

Figure 7.7: MNE plots with high-pass filtering

As one can see from the plots in Figure 7.7, the drifts in the measurements are too big to be fixed by filtering.

fNIRS

The analysis of fNIRS data is first executed by reading the XDF file. However, a lot of preprocessing of the data is needed in order for the MNE tool to be utilized. For example, the channels must be ordered in haemoglobin pairs, such that for a single channel all the types are in subsequent indices. The type order must be 'hbo' then 'hbr'[1]. Therefore, an analysis of the fNIRS data provided by the NirStar software is used to perform the other methods. Figure 7.8 shows the MNE's plot of fNIRS measurements, which has the same outline as the fNIRS plot in Figure 7.1. This indicates that the conversion was successful.

[1]https://mne.tools/stable/auto$_t utorials/io/30_r eading_f nirs_d ata.htmlsphx - glr - auto - tutorials - io - 30 - reading - fnirs - data - py$

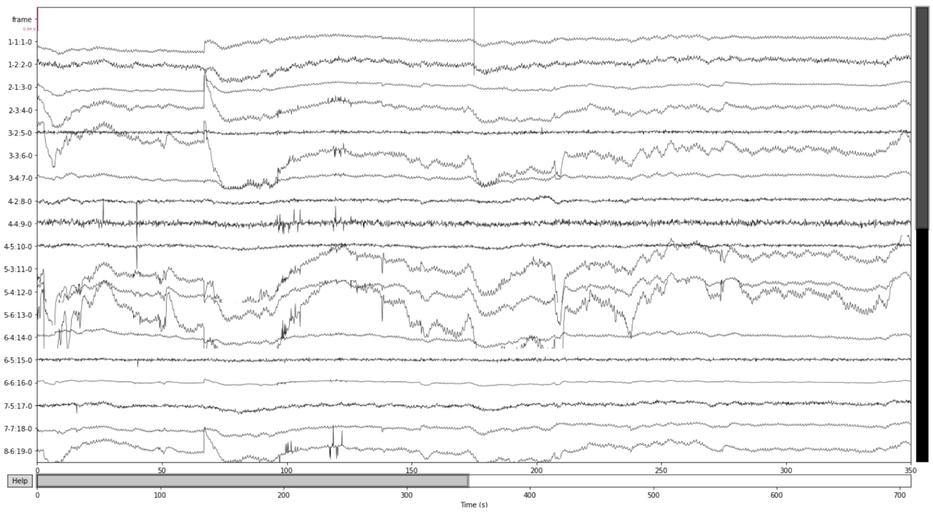


Figure 7.8: MNE plot of fNIRS measurements

Figure 7.9 shows all channels that are not considered to be too close together (short channels).

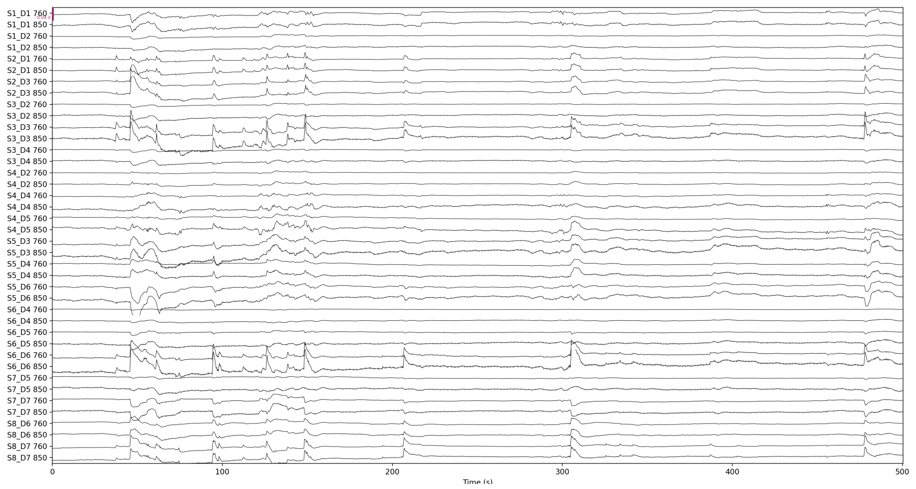


Figure 7.9: MNE plot of appropriate channels for detecting neural responses

Figure 7.10 shows the values after the raw intensity values has been converted to optical density. From optical density the values are again converted to haemoglobin concentration using the modified Beer-Lambert law, Figure 7.11.

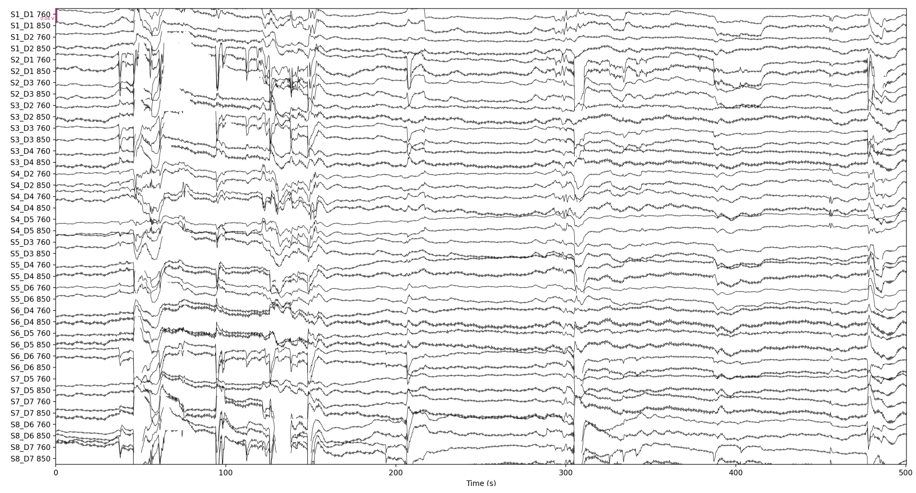


Figure 7.10: MNE plot of optical density

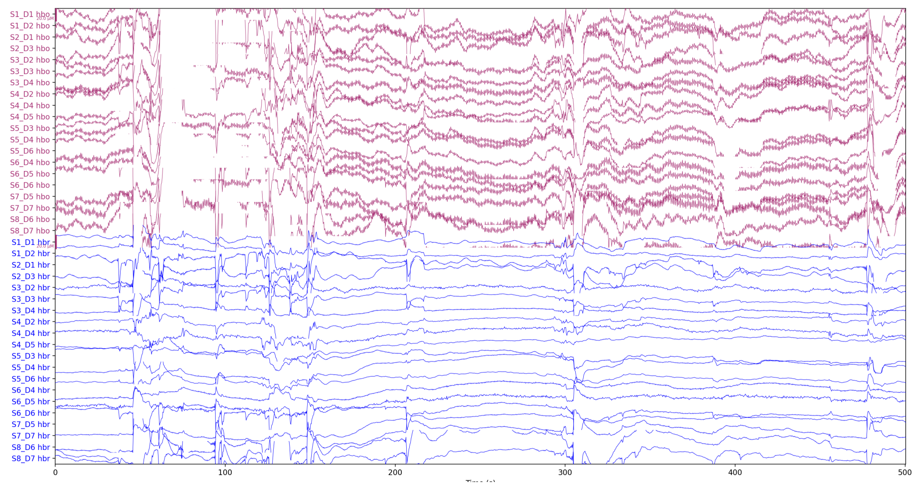


Figure 7.11: MNE plot of haemoglobin concentrations using the modified Beer-Lambert law

7.3 GitHub

The implementation of code can be found in the GitHub repository: **Multimodal-sensor-setup**. The original repository included the recorded measurement data from the participants. Due to the publication in NordDesign conference, a new repository was created exclude the participants' data.

Chapter 8

Demonstration

This thesis aimed to create a low-cost, portable experimental sensor setup featuring multimodal neuroimaging utilizing open-source. As most neuroimaging studies are conducted in laboratories (Goucher-Lambert et al., 2019), there is a need for a portable experiment setup as well as a need for a demonstration in situ. This chapter demonstrates the feasibility of the setup, by testing a real-world use case. The multimodal setup was placed into a backpack and taken for a walk around campus, see Figure 8.1. The measurement data from the various sensors are presented together with lessons learned, which include acceptable and unacceptable artifacts, data quality, and other scenarios in which the setup can be used.

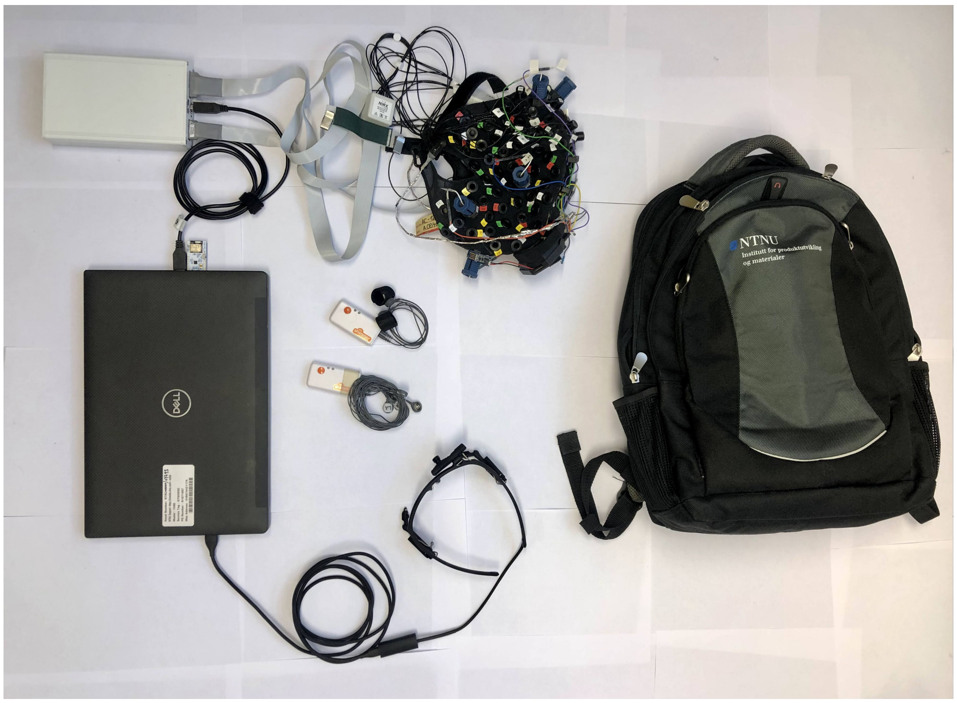


Figure 8.1: The complete experiment setup for the in situ experiment. The laptop and fNIRS device (white box) were placed in the backpack, the ECG in the participant's front pocket, and the GSR in their hand.

The demonstration was performed to investigate how the sensors would perform while walking. Body movement is a known artifact of both eye-tracking, EEG, ECG, and GSR, however, the threshold for the given sensors is not known. The world-view camera of the eye-tracker recorded the entire session. The sensors were placed on the participant, according to the protocol described earlier in this thesis. After initiating the data streams the laptop together with the fNIRS station, was placed in a backpack and put on the participant. The session was timed, ensuring the recorded data was within the experiment time frame, excluding the time used to place the equipment in and out of the backpack. The participant was asked to follow vocal directions from the experimenter. Prior to the experiment, there were several hypotheses on how the sensors would react.

1. Brain waves will be slower when closing the eyes.
2. fNIRS should not be affected by walking
3. Walking up the stairs will increase heart rate and galvanic response, due to body movements
4. EEG will be affected by body movements
5. Places with more stimuli (many people) will increase brain activity compared to walking in corridors.

8.1 Demonstration outline

The participant started the session by sitting on a chair, calmly for one minute, in order to find the baseline for the sensors. Next, he was asked to close his eyes for 30 seconds. According to literature (Antonenko et al., 2010; Balters & Steinert, 2017), it is stated that the EEG can measure changes in the brain waves whether the eyes are open or closed. Given the electrical impulses are faster than the hemodynamic response, the time interval was set by the condition of the fNIRS, which indicates that the hemodynamic response goes back to baseline after approximately 16 seconds. 30 seconds was therefore viewed as fitting. After the eyes were opened, the participant was asked to stand up and leave the room. A walk down the hall, and up the stairs two floors was executed next, in order to increase the heart rate and see how that might have affected both fNIRS, ECG, and GSR. The participant was asked to sit down on a bench at the top of the stairs, to relax for one minute, and for the heart rate to find its way back to baseline. Next, the participant walked down the stairs three floors, before walking into a larger area, where other students socialized. The area was included to see if more stimuli affected the cognitive response. The walk continued through the halls of campus, before reaching the elevator. Introducing a new stimulus, to see how the participant reacted to small spaces. After reaching the intended floor, the participant walked back to the room where it all started and sat down once again in the chair. After one minute of relaxation in the chair, the experiment was over.

The entire session is captured by the world camera on the eye tracker, and can be viewed by following this link: **A walk around campus**.

8.2 Analysis of the data

The description above includes why the participant was asked to do the named actions. The stated hypothesis are based on the theory and data processing considerations presented in this thesis. To ensure the setup responds the way it is intended, the hypothesis should be fulfilled.

Analyzing data from an in situ experiment is difficult due to all stimuli the participant is exposed to. All sensor measurements will include different kinds of noise, which is difficult to detect. The advantage of including an eye-tracker in the experiment setup is that it captures the participant's viewpoint allowing the researcher to eliminate, to some degree, various ERPs. In addition, by correlating blinking from the eye-tracker with the EEG data, it is possible to reduce the distortion that eye movement contributes to.

The outcome of the measurements of the various sensors is given in Figure 8.2. The experiment started at 11750, according to the plot in Figure 8.2, when all equipment was placed in the backpack and the participant was sitting in the chair. As one can see, the measurements until the start point of the experiment are not processable. As expected, there are huge differences captured by the EEG between walking and sitting down. In this case, removing eye movements will not increase the quality of the EEG data, as there is too much noise from other body movements. It is interesting that the GSR increases during relaxation. Measurements captured by the fNIRS show that body movements are not affecting the data, at least not to the same extent as EEG.

The overall quality of the EEG data is poor. The drop in measurement in several of the electrodes indicates sources of error. The cause of the noise is difficult to determine, however it is most likely caused by body movements, which to some extent can be confirmed by the accelerometer measurements. Other observations from the EEG measurements indicate that body movement results in quicker oscillations compared to relaxing in a chair. The findings confirm relevant literature and theory, which says that brainwaves of high frequency correspond to alertness and activity (Abhang et al., 2016).

Observations of the fNIRS data show an increase in oscillations during body movements as well. The higher frequency of oscillations may be caused by neural activity, or it may be due to noise caused by an increase in blood flow due to body movements. As mentioned in the data processing section, an increase in blood flow will affect the oxygen levels in the brain, which in turn affects the hemodynamic response. The abrupt spikes indicate most likely movement in the optode position, causing a short breach in the data. As one can see, the abrupt spikes are more likely to occur during movement compared to sitting still.

The intention of the demonstration was to show how the setup handles in situ experiment situations. Further statistical analyses will not be executed, and are considered outside the scope of this thesis.

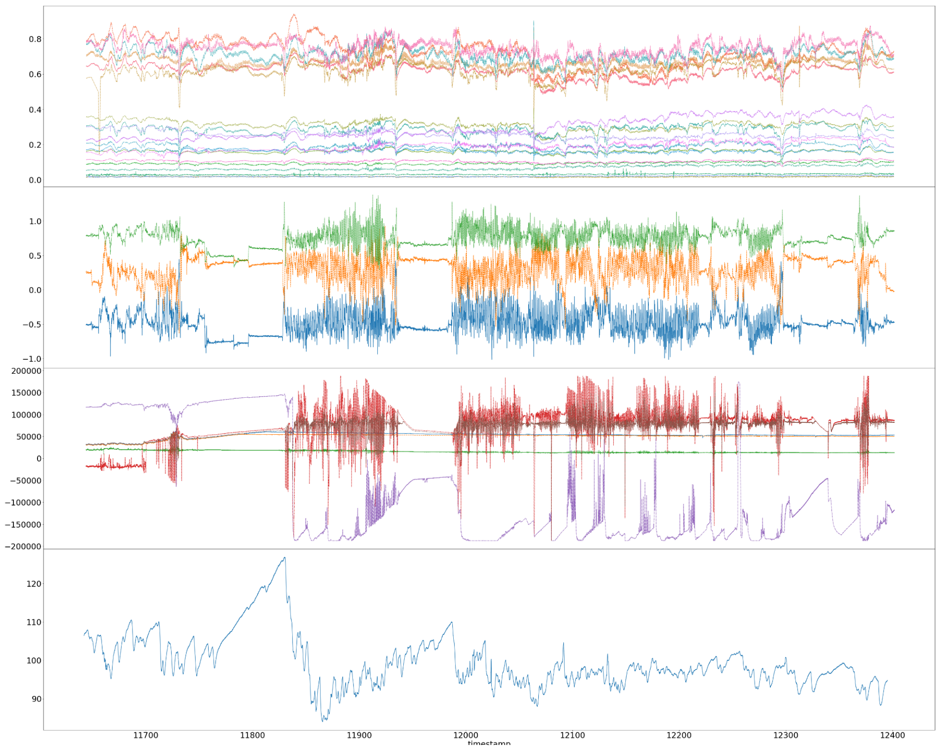


Figure 8.2: All sensors synchronized to the same timestamp, with fNIRS displayed at the top, then EEG_AUX, EEG, and lastly GSR

8.3 Lessons learned

The technological constraint of body movements is still a concern and should be addressed in further research. It is highly valuable to measure the brain activity associated with EEG, even when the participant is moving. However, today there is no alternative to EEG or improvements of the sensor that eliminates noise from body movements.

Chapter 9

Discussion

The primary goal of the project described in this thesis was to develop a low-cost multimodal sensor setup with the use of open-source software distributions. The advantages of utilizing open-source options versus proprietary are many and are clearly discussed in the article provided in chapter 4. The experiment setup facilitates many possibilities, however, there are still challenges arising from the technology itself.

Experiment procedures arising from the psychology field are being adopted by the human-computer interaction field. Controlled, laboratory settings are a standard way of conducting experiments, especially because the equipment used to execute the experiment usually is not mobile. Neuroimaging measurements are commonly conducted by an fMRI machine, which is not only expensive but highly restrictive as well. The development of mobile sensors is only the last couple of decades beginning to emerge, opening up possibilities outside a laboratory setting. New cognitive neuroscience techniques contribute, such as EEG and fNIRS, to increase the availability of cognitive measurement methods which can substitute the gold standard within the field - fMRI.

It is crucial to understand the cognitive processes in interaction with technical systems. Human interaction with computers and machines has increased drastically since the debut of technical systems, and operative tasks are constantly being automated. Interfaces of machines, computers, and other technical systems are becoming increasingly complex, which increases the demand to evaluate the interfaces and the style of interaction (Blandford et al., 2008). Understanding how people use, experience, perceive and process interactive, and increasingly complex technology is best done in the environment of where the technology is being used. According to one study, mentally simulated tasks do not elicit the same brain activation as their real-world counterparts (Okamoto et al., 2004). Thus, a need for a mobile experiment setup that can be used in the desired context is needed.

A multimodal experiment setup provides several advantages. First and foremost it promotes high-quality research by applying data triangulation. In situ experiments, especially, require multiple sensors due to the many sources of error that can interfere with the measurements (Balters & Steinert, 2017). For example, combining the technological principles of EEG and fNIRS provides different approaches to the same constructs, resulting in more information about the human cognition (Ahn & Jun, 2017). Synchronizing the recorded data allows researchers to confirm responses to stimuli, via data triangulation, and detect noise as a source of error. In addition, data triangulation takes into account individual differences (i.e. physiology, psychology, and behaviour) rather than assuming a generalized participant (Balters & Steinert, 2017; Steinert & Jablokow, 2013). The experiment setup integrates neuroimaging modalities, EEG and fNIRS, with systemic measures, ECG and GSR, as well as behaviour using eye-tracking and video recordings. By including all modalities into one setup it allows for an encompassing view of how neurodynamics is coordinated with other systemic changes. This enables research at the interface of emotion and cognition, such as anxiety and stress (Pinti et al., 2020).

The current options system integration of multimodal setups are limited. The advances within neuroimaging technologies have resulted in increasingly wearable sensors, such as the ones used in this setup, developed by OpenBCI and NirSport (Pinti et al., 2020).

The sensors used in this setup are all well known within the HCI field and are considered natural to include in HCI experiments, which are exemplified in the theory chapter. Most experiments are restricted to one sensor, due to technological constraints. Some combinations of sensors are more common than others. EEG and fNIRS, for example, are becoming an increasingly used setup combination (Dybvik, Erichsen, et al., 2021; Khan et al., 2014). Another existing combination is fNIRS and eye-tracking, in a study aimed to investigate the usability of a new web-based investment tool (Bhatt et al., 2019).

Currently, multimodal data capture systems are limited which may be the reason why there are not too many studies including multiple sensors (Ahn & Jun, 2017). One solution used by Ahn et al. (2016) is custom-built and requires two desktop computers. Another approach to multimodal data capturing includes separate instruments which record data independently before post-experiment data synchronization (Al-Shargie et al., 2016).

Mobile physiological sensors are not just providing valuable in tel for engineers and product development. Within the field of health care, wearable sensors together with artificial intelligence (AI) are contributing to revolutionary changes (Chankova, 2022b). The use of wearable sensors, such as smart wristbands, watches, rings, and patches is set to reshape health care in three ways. First, they will detect diagnosis early, leading to less severe disease and cheaper treatment. Second, it will be easier to personalize treatment and third, manage chronic diseases (Chankova, 2022b). Just like in situ HCI experiments provide valuable information about the human response in the desired context, monitoring patients in their natural habitat allows researchers to see how patients experience a given disease and treatment (Chankova, 2022a). In addition, continuous monitoring of patients with chronic conditions can improve their treatment and outcomes significantly. Furthermore, wearable sensors enable doctors to treat more patients, reducing the demand for specialists in areas where there aren't enough (Chankova, 2022a).

The multimodal sensor setup described in this thesis provides researchers with several opportunities. The utilization of LSL facilitates a highly modular setup, allowing many suitable sensors to be included. All sensors used in this setup are portable. It is possible to transport the complete setup with a standard backpack, which was demonstrated by the walk around campus. As the setup is applicable for in situ studies, only the imagination sets the boundaries of which experiments to conduct. Nonetheless, the limitations of the sensors must be taken into consideration as there are multiple sources of error that can affect the data measurements. Moreover, utilizing the LSL and all its features, especially the synchronization of data streams, is a huge advantage in the data analysis process.

Even though the experiment setup is a low-cost solution to conduct multimodal sensory experiments, utilizing open-source distributions comes with a small disadvantage. Given the nature of open-source, the solutions provided are generally developed to handle most use cases. Thus, not all scenarios are accounted for, which is clearly demonstrated in the experiment setup. For example, the EEG sensor needs to be calibrated before the data stream can be sent to LSL. Since LSL does not provide a way of calibrating the signal, another software must be used - OpenBCI. The same goes for the eye-tracker and fNIRS, which must be calibrated through their own software. This results in the setup containing several software and custom scripts, which decreases the setup's usability. The flow chart in Figure 5.2, visualises how all modalities streams and connect to LSL.

All components of the experiment setup are publicly available and can be recreated, however, given the multidisciplinary aspect of the setup it can be difficult. The integration of EEG and fNIRS require associated knowledge and may prove difficult to recreate. Furthermore, various components of the setup are tailored to TrollLABS's requirements and preferences. Even though all configuration is provided, a certain level of knowledge about the various sensors, in addition to programming and engineering skills, are needed to replicate the setup. In spite of the low-cost solutions provided by the setup, there may be barriers to using it, including a lack of resources such as knowledge and time.

Another limitation of the sensor setup concerns the integration of fNIRS and EEG. Even though the devices allow simultaneous measurements of EEG and fNIRS, it is difficult to record neural activity from the same location (Ahn & Jun, 2017). In addition, fNIRS measures temporal features but EEG measures both temporal and spectral. Thus, experiments conducted with EEG normally include rapid and short stimuli. Such stimuli are not applicable in fNIRS recordings, because of the delay of the hemodynamic response (Ahn & Jun, 2017). Therefore, it is important to consider the temporal synchronization of fNIRS and EEG in relation to the different measurement delays and temporal resolutions. Using fNIRS features as prior information to estimate cortical current in EEG is one computational method for dealing with the temporal synchronization issue. Another computational method is to normalize the features of fNIRS and EEG that range from 0 to 1 and apply a summation of the features. Yet another method is to aggregate normalized features of fNIRS and EEG (Ahn & Jun, 2017).

Pilot experiments involve some uncertainty regarding the results. As the aim of the thesis was to incorporate EEG, fNIRS, ECG, GSR, and eye tracker into one mobile setup, it was essential that the laptop could withstand the workload of the data capturing. The results reflect that the goal was accomplished. However, not all sensors provide measurements of high quality. This goes especially for the EEG. As the analysis tool, MNE visualizes in figure Figure 7.7, correcting drift was not possible in this case, due to poor quality. By following a better application protocol, the quality of especially the EEG will improve significantly. Another uncertainty is the conversion of ECG data to LSL. As mentioned in the results, the signal wave presented deviates from the normal waveform. This is most likely due to a calculation error or misconfiguration associated with the conversion of raw ECG signal to LSL stream. Despite thorough research, the configurations and calculations made by both iMotions and Texas Instruments have proved difficult to be found.

Thus, high-quality data measurements from ECG are not ensured. In order to retrieve high-quality data from the ECG, the research in finding the right configurations should continue and is thus included in further work.

An assessment of the results from the analysis tools used in this thesis is not included. Evaluating what the responses indicate, is not within the scope of the thesis as it passes into another discipline. Nonetheless, the results show that the setup may be utilized to perform advanced analysis utilizing open-source tools.

Chapter 10

Conclusion and further work

The goals of this thesis were accomplished, as the result is a functional multimodal sensor setup that primarily utilizes open-source software. The setup's capabilities have been demonstrated through an Ideation pilot experiment, and in situ during a walk around campus.

The data capturing is easily handled by LSL, where all sensors are recorded synchronously, with the possibility of adding more modalities to the setup. Further, the data can be used to perform statistical analysis. The open-source analysis tools used in this thesis provide results in agreement with relevant literature. Thus, this sensor setup represents a reliable method of capturing neurophysiological data during a human-machine interaction experiment. The use of technology and new machines will only increase in the coming years, and complex user interfaces must be addressed in the situation it is used. There is a need for portable experiment setups, and this thesis addresses this need by offering a portable setup that can be used in unpredictable, real-life contexts, which is desired in affective engineering.

The development of the setup resulted in an article that has been accepted for publication at the NordDesign Conference 2022. It contributes to open research, in line with NTNU's policy, and pushes the boundaries of multimodal data capturing.

10.1 Further work

In order to ensure good quality data and analysis, all experiments should include a sequence of known stimuli which is known to trigger cognitive load. In that way, it is possible to capture each individual response pattern, especially how long it takes for the hemodynamic response to show, given the latency it has. The researcher should have clarity in the individual response pattern before experimenting with new stimuli, from which to draw new theories.

In situ experiments face a challenge when using eye-tracking. It is difficult to analyze the videos recorded from both the world camera and the eye camera. Compared to experiments where the participant looks at a screen, with factors already accounted for such as distance from screen and head position, these factors are not known in in situ experiments. Heat maps or other statistical analysis will not be possible to carry out, as the position of the participant is not known. In order to accommodate the missing features, a 3D model of the in situ experiment location should be made. In that way, it is possible to track the position of the head which in turn locates the gaze of the participant.

In addition, the calculation issue in the custom script concerning the ECG should be addressed.

Bibliography

Abelson, F. G. (2021). Integration of Mobile Physiological Sensors For Use in Pilot Experiments [Accepted: 2021-12-01T18:19:34Z Publisher: NTNU]. Retrieved May 2, 2022, from https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2832446

Abhang, P. A., Gawali, B. W., & Mehrotra, S. C. (2016). Chapter 2 - Technological Basics of EEG Recording and Operation of Apparatus. In P. A. Abhang, B. W. Gawali, & S. C. Mehrotra (Eds.), *Introduction to EEG- and Speech-Based Emotion Recognition* (pp. 19–50). Academic Press. https://doi.org/10.1016/B978-0-12-804490-2.00002-6

Ahn, S., & Jun, S. C. (2017). Multi-Modal Integration of EEG-fNIRS for Brain-Computer Interfaces – Current Limitations and Future Directions. *Frontiers in Human Neuroscience*, *11*. Retrieved May 26, 2022, from https://www.frontiersin.org/article/10.3389/fnhum.2017.00503

Ahn, S., Nguyen, T., Jang, H., Kim, J. G., & Jun, S. C. (2016). Exploring Neuro-Physiological Correlates of Drivers' Mental Fatigue Caused by Sleep Deprivation Using Simultaneous EEG, ECG, and fNIRS Data. *Frontiers in Human Neuroscience*, *10*. Retrieved June 10, 2022, from https://www.frontiersin.org/article/10.3389/fnhum.2016.00219

Al-Shargie, F., Kiguchi, M., Badruddin, N., Dass, S. C., Hani, A. F. M., & Tang, T. B. (2016). Mental stress assessment using simultaneous measurement of EEG and fNIRS [Publisher: Optica Publishing Group]. *Biomedical Optics Express*, *7*(10), 3882–3898. https://doi.org/10.1364/BOE.7.003882

Antonenko, P., Paas, F., Grabner, R., & van Gog, T. (2010). Using Electroencephalography to Measure Cognitive Load. *Educational Psychology Review*, *22*(4), 425–438. https://doi.org/10.1007/s10648-010-9130-y

Atsumori, H., Kiguchi, M., Katura, T., Funane, T., Obata, A., Sato, H., Manaka, T., Iwamoto, M., Maki, A., Koizumi, H., & Kubota, K. (2010). Noninvasive imaging of prefrontal activation during attention-demanding tasks performed while walking using a wearable optical topography system [Publisher: SPIE]. *Journal of Biomedical Optics*, *15*(4), 046002. https://doi.org/10.1117/1.3462996

Balardin, J. B., Zimeo Morais, G. A., Furucho, R. A., Trambaiolli, L., Vanzella, P., Biazoli, C., & Sato, J. R. (2017). Imaging Brain Function with Functional Near-Infrared Spectroscopy in Unconstrained Environments. *Frontiers in Human Neuroscience*, *11*. Retrieved June 1, 2022, from https://www.frontiersin.org/article/10.3389/fnhum.2017.00258

Balters, S., & Steinert, M. (2017). Capturing emotion reactivity through physiology measurement as a foundation for affective engineering in engineering design science and engineering practices. *Journal of Intelligent Manufacturing*, *28*(7), 1585–1607. https://doi.org/10.1007/s10845-015-1145-2

Başar, E., Başar-Eroğlu, C., Karakaş, S., & Schürmann, M. (2000). Brain oscillations in perception and memory. *International Journal of Psychophysiology*, *35*(2), 95–124. https://doi.org/10.1016/S0167-8760(99)00047-1

Benedek, M., & Kaernbach, C. (2010). A continuous measure of phasic electrodermal activity. *Journal of Neuroscience Methods*, *190*(1), 80–91. https://doi.org/10.1016/j.jneumeth.2010.04.028

Bhatt, S., Agrali, A., McCarthy, K., Suri, R., & Ayaz, H. (2019). Chapter 30 - Web Usability Testing With Concurrent fNIRS and Eye Tracking. In H. Ayaz & F. Dehais (Eds.), *Neuroergonomics* (pp. 181–186). Academic Press. https://doi.org/10.1016/B978-0-12-811926-6.00030-0

Blandford, A., Cox, A. L., & Cairns, P. (2008). Controlled experiments. In A. L. Cox & P. Cairns (Eds.), *Research Methods for Human-Computer Interaction* (pp. 1–16). Cambridge University Press. https://doi.org/10.1017/CBO9780511814570.002

Blessing, L. T., & Chakrabarti, A. (Eds.). (2009). DRM: A Design Reseach Methodology. In *DRM, a Design Research Methodology* (pp. 13–42). Springer. https://doi.org/10.1007/978-1-84882-587-1_2

Blignaut, P., & Wium, D. (2014). Eye-tracking data quality as affected by ethnicity and experimental design. *Behavior Research Methods*, *46*(1), 67–80. https://doi.org/10.3758/s13428-013-0343-0

Cardoso, C., & Badke-Schaub, P. (2011). The Influence of Different Pictorial Representations During Idea Generation [_eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/j.2162-6057.2011.tb01092.x]. *The Journal of Creative Behavior*, *45*(2), 130–146. https://doi.org/10.1002/j.2162-6057.2011.tb01092.x

Carter, B. T., & Luke, S. G. (2020). Best practices in eye tracking research. *International Journal of Psychophysiology*, *155*, 49–62. https://doi.org/10.1016/j.ijpsycho.2020.05.010

Chankova, S. (2022a). Data from wearable devices are changing disease surveillance and medical research. *The Economist*. Retrieved June 9, 2022, from https://www.economist.com/technology-quarterly/2022/05/02/data-from-wearable-devices-are-changing-disease-surveillance-and-medical-research

Chankova, S. (2022b). Wearable technology promises to revolutionise health care. *The Economist*. Retrieved June 9, 2022, from https://www.economist.com/leaders/2022/05/05/wearable-technology-promises-to-revolutionise-health-care

Croft, R. J., & Barry, R. J. (2000). Removal of ocular artifact from the EEG: A review. *Neurophysiologie Clinique/Clinical Neurophysiology*, *30*(1), 5–19. https://doi.org/10.1016/S0987-7053(00)00055-1

Cui, X., Bray, S., Bryant, D. M., Glover, G. H., & Reiss, A. L. (2011). A quantitative comparison of NIRS and fMRI across multiple cognitive tasks. *NeuroImage*, *54*(4), 2808–2821. https://doi.org/10.1016/j.neuroimage.2010.10.069

Dix, A., Finlay, J., Abowd, G. D., & Beale, R. (2004). *Human-computer interaction*. Pearson Education.

Duchowski, A. T. (2002). A breadth-first survey of eye-tracking applications. *Behavior Research Methods, Instruments, & Computers*, *34*(4), 455–470. https://doi.org/10.3758/BF03195475

Dybvik, H., Erichsen, C. K., & Steinert, M. (2021). Demonstrating the Feasibility of Multimodal Neuroimaging Data Capture with a Wearable Electoencephalography + Functional Near-Infrared Spectroscopy (EEG+FNIRS) in Situ. *Proceedings of the Design Society*, *1*, 901–910. https://doi.org/10.1017/pds.2021.90

Dybvik, H., Kuster Erichsen, C., & Steinert, M. (2021). Description of a Wearable Electroencephalography + Functional Near-Infrared Spectroscopy (EEG+FNIRS) for In-Situ Experiments on Design Cognition. *Proceedings of the Design Society*, *1*, 943–952. https://doi.org/10.1017/pds.2021.94

Erichsen, C. K., Dybvik, H., & Steinert, M. (2020). Integration of low-cost, dry-comb EEG-electrodes with a standard electrode cap for multimodal signal acquisition during human experiments. *DS 101: Proceedings of NordDesign 2020, Lyngby, Denmark, 12th - 14th August 2020*, 1–12. https://doi.org/10.35199/NORDDESIGN2020.19

Extensible Data Format (XDF) [original-date: 2015-07-21T19:59:41Z]. (2021). Retrieved November 20, 2021, from https://github.com/sccn/xdf

Flink, R., Pedersen, B., Guekht, A. B., Malmgren, K., Michelucci, R., Neville, B., Pinto, F., Stephani, U., & Özkara, C. (2002). Guidelines for the use of EEG methodology in the diagnosis of epilepsy [_eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1034/j.1600-0404.2002.01361.x]. *Acta Neurologica Scandinavica*, *106*(1), 1–7. https://doi.org/10.1034/j.1600-0404.2002.01361.x

Fu, K., Chan, J., Cagan, J., Kotovsky, K., Schunn, C., & Wood, K. (2013). The meaning of "near" and "far": The impact of structuring design databases and the effect of distance of analogy on design output [Publisher: American Society of Mechanical Engineers Digital Collection]. *Journal of Mechanical Design*, *135*(2).

Gacek, A., & Pedrycz, W. (2011). *ECG Signal Processing, Classification and Interpretation: A Comprehensive Framework of Computational Intelligence* [Google-Books-ID: lPTiGqPKY94C]. Springer Science & Business Media.

Goldschmidt, G., & Smolkov, M. (2006). Variances in the impact of visual stimuli on design problem solving performance. *Design Studies*, *27*(5), 549–569. https://doi.org/10.1016/j.destud.2006.01.002

Goucher-Lambert, K., & Cagan, J. (2017). Using crowdsourcing to provide analogies for designer ideation in a cognitive study, 529–538.

Goucher-Lambert, K., Moss, J., & Cagan, J. (2019). A neuroimaging investigation of design ideation with and without inspirational stimuli—understanding the meaning of near and far stimuli. *Design Studies*, *60*, 1–38. https://doi.org/10.1016/j.destud.2018.07.001

Gramfort, A., Luessi, M., Larson, E., Engemann, D., Strohmeier, D., Brodbeck, C., Goj, R., Jas, M., Brooks, T., Parkkonen, L., & Hämäläinen, M. (2013). MEG and EEG data analysis with MNE-Python. *Frontiers in Neuroscience*, *7*. Retrieved May 31, 2022, from https://www.frontiersin.org/article/10.3389/fnins.2013.00267

Hay, L., Cash, P., & McKilligan, S. (2020). The future of design cognition analysis [Publisher: Cambridge University Press]. *Design Science*, *6*. https://doi.org/10.1017/dsj.2020.20

Hocke, L. M., Oni, I. K., Duszynski, C. C., Corrigan, A. V., Frederick, B. D., & Dunn, J. F. (2018). Automated Processing of fNIRS Data—A Visual Guide to the Pitfalls and Consequences [Number: 5 Publisher: Multidisciplinary Digital Publishing Institute]. *Algorithms*, *11*(5), 67. https://doi.org/10.3390/a11050067

iMotions. (2017). GSR Pocket Guide.

Jansson, D. G., & Smith, S. M. (1991). Design fixation. *Design Studies*, *12*(1), 3–11. https://doi.org/10.1016/0142-694X(91)90003-F

Just, M. A., & Carpenter, P. A. (n.d.). A theory of reading: From eye fixations to comprehension. [Publisher: US: American Psychological Association]. *Psychological Review*, *87*(4), 329. https://doi.org/10.1037/0033-295X.87.4.329

Kassner, M., Patera, W., & Bulling, A. (2014). Pupil: An open source platform for pervasive eye tracking and mobile gaze-based interaction. *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication*, 1151–1160. https://doi.org/10.1145/2638728.2641695

Khan, M. J., Hong, M. J., & Hong, K.-S. (2014). Decoding of four movement directions using hybrid NIRS-EEG brain-computer interface. *Frontiers in Human Neuroscience*, *8*. Retrieved June 10, 2022, from https://www.frontiersin.org/article/10.3389/fnhum.2014.00244

Kothe, C., Medine, D., Boulay, C., Grivich, M., & Stenner, T. (2019a). Introduction — Labstreaminglayer. Retrieved May 6, 2022, from https://labstreaminglayer.readthedocs.io/info/intro.html

Kothe, C., Medine, D., Boulay, C., Grivich, M., & Stenner, T. (2019b). Quick Start — Labstreaminglayer. Retrieved October 8, 2021, from https://labstreaminglayer.readthedocs.io/info/getting_started.html

Labs, P. (2022). Core - Best Practices. Retrieved May 11, 2022, from https://docs.pupil-labs.com

Ledowski, T., Bromilow, J., Wu, J., Paech, M. J., Storm, H., & Schug, S. A. (2007). The assessment of postoperative pain by monitoring skin conductance: Results of a prospective study* [_eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1365-2044.2007.05191.x]. *Anaesthesia*, *62*(10), 989–993. https://doi.org/10.1111/j.1365-2044.2007.05191.x

Leikanger, K. K. (2016). Prototyping an Experimental Setup for Understanding Affective Response in a Ship Bridge Scenario [Accepted: 2016-04-14T14:00:26Z Publisher: NTNU]. *100*. Retrieved May 6, 2022, from https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2385692

Linsey, J. S., Wood, K. L., & Markman, A. B. (2008). Modality and representation in analogy [Publisher: Cambridge University Press]. *AI EDAM*, *22*(2), 85–100. https://doi.org/10.1017/S0890060408000061

Linsey, J. S., Markman, A. B., & Wood, K. L. (2012). Design by analogy: A study of the WordTree method for problem re-representation.

Linsey, J. S., & Viswanathan, V. K. (2014). Overcoming Cognitive Challenges in Bioinspired Design and Analogy. In A. K. Goel, D. A. McAdams, & R. B. Stone (Eds.), *Biologically Inspired Design: Computational Methods and Tools* (pp. 221–244). Springer. https://doi.org/10.1007/978-1-4471-5248-4_9

Makowski, D., Pham, T., Lau, Z. J., Brammer, J. C., Lespinasse, F., Pham, H., Schölzel, C., & Chen, S. H. A. (2021). NeuroKit2: A Python toolbox for neurophysiological signal processing. *Behavior Research Methods*, *53*(4), 1689–1696. https://doi.org/10.3758/s13428-020-01516-y

Malmivuo, J., & Plonsey, R. (1995). Bioelectromagnetism. 19. The Basis of ECG Diagnosis.

Miller, S. R., Bailey, B. P., & Kirlik, A. (2014). Exploring the Utility of Bayesian Truth Serum for Assessing Design Knowledge [Publisher: Taylor & Francis]. *Human–Computer Interaction*, *29*(5-6), 487–515. https://doi.org/10.1080/07370024.2013.870393

Nagai, Y., Jones, C. I., & Sen, A. (2019). Galvanic Skin Response (GSR)/Electrodermal/Skin Conductance Biofeedback on Epilepsy: A Systematic Review and Meta-Analysis. *Frontiers in Neurology*, *10*. Retrieved May 24, 2022, from https://www.frontiersin.org/article/10.3389/fneur.2019.00377

Nosek, B. A., Alter, G., Banks, G. C., Borsboom, D., Bowman, S. D., Breckler, S. J., Buck, S., Chambers, C. D., Chin, G., Christensen, G., Contestabile, M., Dafoe, A., Eich, E., Freese, J., Glennerster, R., Goroff, D., Green, D. P., Hesse, B., Humphreys, M., . . . Yarkoni, T. (2015). Promoting an open research culture [Publisher: American Association for the

Advancement of Science]. *Science*, *348*(6242), 1422–1425. https://doi.org/10.1126/science.aab2374

Okamoto, M., Dan, H., Shimizu, K., Takeo, K., Amita, T., Oda, I., Konishi, I., Sakamoto, K., Isobe, S., Suzuki, T., Kohyama, K., & Dan, I. (2004). Multimodal assessment of cortical activation during apple peeling by NIRS and fMRI. *NeuroImage*, *21*(4), 1275–1288. https://doi.org/10.1016/j.neuroimage.2003.12.003

Oostenveld, R., & Praamstra, P. (2001). The five percent electrode system for high-resolution EEG and ERP measurements. *Clinical Neurophysiology*, *112*(4), 713–719. https://doi.org/10.1016/S1388-2457(00)00527-7

Overview [original-date: 2018-02-28T12:17:09Z]. (2021). Retrieved November 20, 2021, from https://github.com/labstreaminglayer/App-LabRecorder

Peirce, J., Gray, J. R., Simpson, S., MacAskill, M., Höchenberger, R., Sogo, H., Kastman, E., & Lindeløv, J. K. (2019). PsychoPy2: Experiments in behavior made easy. *Behavior Research Methods*, *51*(1), 195–203. https://doi.org/10.3758/s13428-018-01193-y

Pinti, P., Tachtsidis, I., Hamilton, A., Hirsch, J., Aichelburg, C., Gilbert, S., & Burgess, P. W. (2020). The present and future use of functional near-infrared spectroscopy (fNIRS) for cognitive neuroscience [_eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/nyas.13948]. *Annals of the New York Academy of Sciences*, *1464*(1), 5–29. https://doi.org/10.1111/nyas.13948

Poldrack, R. A. (2006). Can cognitive processes be inferred from neuroimaging data? *Trends in Cognitive Sciences*, *10*(2), 59–63. https://doi.org/10.1016/j.tics.2005.12.004

Pourmohammadi, S., & Maleki, A. (2020). Stress detection using ECG and EMG signals: A comprehensive study. *Computer Methods and Programs in Biomedicine*, *193*, 105482. https://doi.org/10.1016/j.cmpb.2020.105482

Purcell, A. T., Williams, P., Gero, J. S., & Colbron, B. (1993). Fixation Effects: Do They Exist in Design Problem Solving? [Publisher: SAGE Publications Ltd STM]. *Environment and Planning B: Planning and Design*, *20*(3), 333–345. https://doi.org/10.1068/b200333

Raine, A., Bihrle, S., Venables, P., Mednick, S., & Pollock, V. (1999). Skin-conductance orienting deficits and increased alcoholism in schizotypal criminals. *Journal of Abnormal Psychology*, *108*(2), 299–306. https://doi.org/10.1037/0021-843X.108.2.299

Research, S. (2017a). Shimmer3 ECG Unit. Retrieved May 24, 2022, from https://shimmersensing.com/product/shimmer3-ecg-unit-2/

Research, S. (2017b). Shimmer3 GSR+ Unit. Retrieved May 24, 2022, from https://shimmersensing.com/product/shimmer3-gsr-unit/

Research, S. (2018). Shimmer ECG USer Guide Revision 1.12.

Shrout, P. E., & Rodgers, J. L. (2018). Psychology, Science, and Knowledge Construction: Broadening Perspectives from the Replication Crisis [_eprint: https://doi.org/10.1146/annurev-psych-122216-011845]. *Annual Review of Psychology*, *69*(1), 487–510. https://doi.org/10.1146/annurev-psych-122216-011845

Smith, S. J. M. (2005). EEG in the diagnosis, classification, and management of patients with epilepsy [Publisher: BMJ Publishing Group Ltd]. *Journal of Neurology, Neurosurgery & Psychiatry*, *76*(suppl 2), ii2–ii7. https://doi.org/10.1136/jnnp.2005.069245

Steinert, M., & Jablokow, K. (2013). Triangulating front end engineering design activities with physiology data and psychological preferences [ISBN: 9781904670506]. *DS 75-7: Proceedings of the 19th International Conference on Engineering Design (ICED13), Design for Harmonies, Vol.7: Human Behaviour in Design, Seoul, Korea, 19-22.08.2013*, 109–118. Retrieved May 2, 2022, from https://www.designsociety.org/publication/34575/Triangulating+front+end+engineering+design+activities+with+physiology+data+and+psychological+preferences

Storm, H. (2008). Changes in skin conductance as a tool to monitor nociceptive stimulation and pain. *Current Opinion in Anesthesiology*, *21*(6), 796–804. https://doi.org/10.1097/ACO.0b013e3283183fe4

Toh, C. A., & Miller, S. R. (2014). The impact of example modality and physical interactions on design creativity [Publisher: American Society of Mechanical Engineers]. *Journal of Mechanical Design*, *136*(9), 091004.

Tseng, I., Moss, J., Cagan, J., & Kotovsky, K. (2008). The role of timing and analogical similarity in the stimulation of idea generation in design. *Design Studies*, *29*(3), 203–221. https://doi.org/10.1016/j.destud.2008.01.003

Viswanathan, V. K., & Linsey, J. S. (2013). Design fixation and its mitigation: A study on the role of expertise [Publisher: American Society of Mechanical Engineers]. *Journal of Mechanical Design*, *135*(5), 051008.

Wilson, J. O., Rosen, D., Nelson, B. A., & Yen, J. (2010). The effects of biological examples in idea generation. *Design Studies*, *31*(2), 169–186. https://doi.org/10.1016/j.destud.2009.10.003

Appendix A

main.py

```python
# TABLE OF CONTENT
#--------------------------------------------------
# 1      Required modules
# 2      Main method
# 2.1    Set comport to GSR, ECG, and EEG
# 2.2    Enable concurrent streaming of data
# 2.3    Start concurrent running of streams


# 1 Required modules
#--------------------
import threading
import time
import random
from ECG_to_LSL import ECG_to_LSL
from EEG_to_LSL import EEG_to_LSL
from GSR_to_LSL import GSR_to_LSL
import sys


# 2 Main method
#------------------
def main():
    # 2.1 Set comport to GSR, ECG, and EEG
    comX = sys.argv[1]      # GSR
    comY = sys.argv[2]      # ECG
    comUSB = 'Com3'         # EEG

    # 2.2 Enable concurrent streaming of data
    gsr = threading.Thread(target=GSR_to_LSL, args=(comX, ))
    ecg = threading.Thread(target=ECG_to_LSL, args=(comY,))
    eeg = threading.Thread(target=EEG_to_LSL, args=(comUSB,))

    # 2.3 Start concurrent running of streams
    gsr.start()
    ecg.start()
    eeg.start()



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

Appendix B

ECG_to_LSL.py

```
# TABLE OF CONTENT
#-----------------------------------------------
# 1         Required modules
# 2         ECG_to_LSL
# 2.1       Initialize ECG sensor
# 2.1.1     Initialize contact with ECG sensor unit
# 2.1.2     Set configuration parameters
# 2.2       Define ECG setup
# 2.2.1     Set Sample rate
# 2.2.2     Request daughter Boa
# 2.2.3     Send Set sensor command
# 2.2.4     Set configuration bytes
# 2.2.5     Set calibration factor
# 2.2.6     Configure Chip1 and Chip2
# 2.2.7     Send start streaming command
# 2.2.8     Define stream info, displayed in LabRecorder
# 2.2.9     Create stream info
# 2.2.10    Create stream outlet
# 2.2.11    Read incoming data and push to LSL stream


# 1 Required modules
#---------------------
import sys, struct, serial, time
from pylsl import StreamInfo, StreamOutlet
from ShimmerCommands import ShimmerCommands

# 2 ECG_to_LSL
#------------------
```

```
class ECG_to_LSL:
   # 2.1 Initialize ECG sensor
    def __init__(self, comX):
        # 2.1.1 Initialize contact with ECG sensor unit
        self.ser = ShimmerCommands.serial_connect(self, comX)
        # 2.1.2 Set configuration parameters
        self.exgconfigGain = {        #decimal
            'GAIN_1':  0x15, #21
            'GAIN_2':  0x25, #37
            'GAIN_3':  0x35, #53
            'GAIN_4':  0x45, #69
            'GAIN_6':  0x05, #5
            'GAIN_8':  0x55, #55
            'GAIN_12': 0x65  #101
        }

        self.exgGain = {
            'GAIN_1':  1, #used
            'GAIN_2':  2,
            'GAIN_3':  3,
            'GAIN_4':  4, #Recommended
            'GAIN_6':  6, #Default
            'GAIN_8':  8,
            'GAIN_12': 12
        }

        self.exg_24bit = [0x18, 0x00, 0x00]
        self.exg_16bit = [0x00, 0x00, 0x18]

        self.samplingFrequency         = 512
         ↪                                                    # frequency in Hz
        self.exgRes_24bit                =
         ↪  False                                                  # 24bit if True,
         ↪  else 16bit
        self.exgGainValue                = self.exgconfigGain['GAIN_1']
         ↪          # sets a gain of 1


        # The internal sampling rate of the ADS1292R chips needs to be set based
         ↪  on the Shimmers sampling rate
        if (self.samplingFrequency<=125):
            self.exgSamplingRate = 0x00 # 125Hz
        elif (self.samplingFrequency<=250):
            self.exgSamplingRate = 0x01 # 250Hz
        elif (self.samplingFrequency<=500):
            self.exgSamplingRate = 0x02 # 500Hz
        elif (self.samplingFrequency<=1000):
            self.exgSamplingRate = 0x03 # 1000Hz
        elif (self.samplingFrequency<=2000):
            self.exgSamplingRate = 0x04 # 2000Hz
        elif (self.samplingFrequency<=4000):
            self.exgSamplingRate = 0x05 # 4000Hz
        else:
            self.exgSamplingRate = 0x02 # 500Hz

        # Chip 1 configuration
        self.chip1Config = [self.exgSamplingRate, 0xA3, 0x10, self.exgGainValue,
        ↪  self.exgGainValue, 0x00, 0x00, 0x00, 0x02, 0x01]

        # Chip 2 configuration
        self.chip2Config = [self.exgSamplingRate, 0xA3, 0x10, self.exgGainValue,
        ↪  self.exgGainValue, 0x00, 0x00, 0x00, 0x02, 0x01]

        self.serNumber = 0
        self.srRev = 0
        self.ECG_setup()

    # 2.2 Define ECG setup

    # 2.2.1 Set Sample rate
    def setSamplingRateHz(self, rate=512):
        # send the set sampling rate command
        sampling_freq = rate #Hz
        clock_wait = (2 << 14) / sampling_freq
        self.ser.write(struct.pack('<BH', 0x05, int(clock_wait)))
        ShimmerCommands.wait_for_ack(self)
        print ("Freq sent...")

    # 2.2.2 Request daughter Board
    def requestDaughterCard(self):
        #get the daughter card ID byte (SR number)
        print("Requesting Daughter Card ID and Revision number...")
        self.ser.write(struct.pack('BBB', 0x66, 0x02,0x00))
        ShimmerCommands.wait_for_ack(self)

        ddata = list(struct.unpack(4*'B', self.ser.read(4)))
        self.srNumber = ddata[2]
        self.srRev = ddata[3]

        print ("Device: SR%d-%d" % (self.srNumber, self.srRev))

    # 2.2.3 Send Set sensor command
    def setSensors(self):
        self.ser.write(struct.pack('BBBB', 0x08, 0x18, 0x00, 0x00))  #exg1 and
        ↪  exg2
        ShimmerCommands.wait_for_ack(self)
        print ("Sensor Enabling done...")

    # 2.2.4 Set configuration bytes
    def setConfigBytes(self):
        if self.exgRes_24bit:
            sensors = [0x08] + self.exg_24bit
        else:
            sensors = [0x08] + self.exg_16bit
        self.ser.write(sensors)
        ShimmerCommands.wait_for_ack(self)
        time.sleep(2)

    # 2.2.5 Set calibration factor
    def  setCalFactor(self):
        if self.exgRes_24bit:
            exgCalFactor = (((2.42*1000)/self.exgGain['GAIN_1'])/(pow(2,23)-1))
        else:
            exgCalFactor =
            ↪  (((2.42*1000)/(self.exgGain['GAIN_1']*2))/(pow(2,15)-1))

        if(self.srNumber == 47 and self.srRev >= 4):
            self.chip1Config[1] |= 8 # Config byte for CHIP1 in SR47-4
        return exgCalFactor

    # 2.2.6 Configure Chip1 and Chip2
    def configureChips(self):
        # Configure Chip 1
        chip1Config = [0x61, 0x00, 0x00, 0x0A] + self.chip1Config
        self.ser.write(chip1Config)
        ShimmerCommands.wait_for_ack(self)

        # Configure Chip 2
        chip2Config = [0x61, 0x01, 0x00, 0x0A] + self.chip2Config
        self.ser.write(chip2Config)
        ShimmerCommands.wait_for_ack(self)
        print ("Configuration sent...")

    # 2.2.7 Send start streaming command
    def startStraming(self):
        # send start streaming command
        self.ser.write(struct.pack('B', 0x07))
        ShimmerCommands.wait_for_ack(self)
        print ("Start sent...")

    def setupLSLStream(self):
        # 2.2.8 Define stream info, displayed in LabRecorder
        name = 'Shimmer_ECG'
        ID = 'Shimmer_Ecg'
        channels = 4
        sample_rate = self.samplingFrequency
        datatype = 'float32'
        streamType = 'ECG'
        print("Creating LSL stream for ECG. \nName: %s\nID: %s\n" %(name, ID))

        # 2.2.9 Create stream info
        info_ecg = StreamInfo(name, streamType, channels, sample_rate, datatype,
        ↪  ID)

        # 2.2.10 Create stream outlet
        chns = info_ecg.desc().append_child("channels")
        for label in ["C1CH1", "C1CH2", "C2CH1", "C2CH2"]:
            ch = chns.append_child("channel")
            ch.append_child_value("label", label)
        outlet_ecg = StreamOutlet(info_ecg)
        return outlet_ecg



    def ECG_setup(self):
        self.requestDaughterCard()
        self.setSensors()
        self.setSamplingRateHz(self.samplingFrequency)
        self.setConfigBytes()
        exgCalFactor = self.setCalFactor()
        self.configureChips()
        self.startStraming()
        outlet_ecg = self.setupLSLStream()

        # 2.2.11 Read incoming data and push to LSL stream
        ddata = ""
        numbytes = 0
        framesize = (18 if self.exgRes_24bit else 14) # 1byte packet type + 3byte
        ↪  timestamp + 14byte ExG data

        print ("Packet Type,\tTimestamp, \tChip1 Status, \tChip1 Channel 1,2
        ↪  (mv), \tChip2 Status, \tChip2 Channel 1,2 (mV)")
        try:
            while True:
                while numbytes < framesize:
                    ddata = self.ser.read(framesize)
                    numbytes = len(ddata)

                data = ddata[0:framesize]
                ddata = ddata[framesize:]
                numbytes = len(ddata)

                (packettype,) = struct.unpack('B', data[0:1])

                (ts0, ts1, ts2, c1status) = struct.unpack('BBBB', data[1:5])

                timestamp = ts0 + ts1*256 + ts2*65536
                # 24-bit signed values MSB values are tricky, as struct only supports 16-bit or 32-bit
                # pad with zeroes at LSB end and then shift the result

                if self.exgRes_24bit:
                    # chip 1
                    c1ch1 = struct.unpack('>i', (data[5:8]))[0] >> 8
                    c1ch2 = struct.unpack('>i', (data[8:11] + '\0'))[0] >> 8

                    # status byte
                    (c2status,) = struct.unpack('B', data[11:12])

                    # chip 2
                    c2ch1 = struct.unpack('>i', (data[12:15] + '\0'))[0] >> 8
                    c2ch2 = struct.unpack('>i', (data[15:framesize] + '\0'))[0] >> 8
                else:
                    # chip 1
                    c1ch1 = struct.unpack('>h', data[5:7])[0]
                    c1ch2 = struct.unpack('>h', data[7:9])[0]

                    # status byte
                    (c2status,) = struct.unpack('B', data[9:10])

                    # chip 2
                    c2ch1 = struct.unpack('>h', data[10:12])[0]
                    c2ch2 = struct.unpack('>h', data[12:framesize])[0]

                # Calibrate exg channels:
                c1ch1 *= exgCalFactor
                c1ch2 *= exgCalFactor
                c2ch1 *= exgCalFactor
                c2ch2 *= exgCalFactor


                ecg_data = [c1ch1, c1ch2, c2ch1, c2ch2]
                ecg_chunk = []
                ecg_chunk.append(ecg_data)
                outlet_ecg.push_chunk(ecg_chunk)
        except: ShimmerCommands.stop_stream(self)
```

Appendix C

GSR_to_LSL.py

```
# TABLE OF CONTENT
#-------------------------------------------------
# 1     Required modules
# 2     GSR_to_LSL
# 2.1   Initialize contact with GSR sensor unit
# 2.2   Define GSR setup
# 2.2.1 Send the set sensors command
# 2.2.2 Enable the internal expansion board power
# 2.2.3 Define stream info, displayed in LabRecorder
# 2.2.4 Create stream info
# 2.2.5 Create stream outlet
# 2.2.6 Send the set sampling rate command
# 2.2.7 Send start streaming command
# 2.2.8 Read incoming data


# 1 Required modules
#---------------------
import struct
from pylsl import StreamInfo, StreamOutlet
from ShimmerCommands import ShimmerCommands

# 2 GSR_to_LSL
#-------------------
class GSR_to_LSL:
   # 2.1 Initialize contact with GSR sensor unit
   def __init__(self, comX):
      self.ser = ShimmerCommands.serial_connect(self, comX)
      self.GSR_setup()
```

```

    # 2.2 Define GSR setup
    def GSR_setup(self):
        # 2.2.1 Send the set sensors command
        self.ser.write(struct.pack('BBBB', 0x08 , 0x04, 0x01, 0x00))  #GSR and PPG
        ShimmerCommands.wait_for_ack(self)
        print( "sensor setting, done.")

        # 2.2.2 Enable the internal expansion board power
        self.ser.write(struct.pack('BB', 0x5E, 0x01))
        ShimmerCommands.wait_for_ack(self)
        print( "enable internal expansion board power, done.")

        # 2.2.3 Define stream info, displayed in LabRecorder
        name = 'Shimmer_GSR'
        ID = 'Shimmer_GSR'
        channels = 1
        sample_rate = 50
        datatype = 'float32'
        streamType = 'GSR'
        print("Creating LSL stream for GSR. \nName: %s\nID: %s\n" %(name, ID))

        # 2.2.4 Create stream info
        info_gsr = StreamInfo(name, streamType, channels, sample_rate, datatype,
         ↪  ID)

        # 2.2.5 Create stream outlet
        chns = info_gsr.desc().append_child("channels")
        ch = chns.append_child("channel")
        ch.append_child_value("label", "CH1")
        outlet_gsr = StreamOutlet(info_gsr)

        # 2.2.6 Send the set sampling rate command
        sampling_freq = 50
        clock_wait = (2 << 14) / sampling_freq
        self.ser.write(struct.pack('<BH', 0x05, int(clock_wait)))
        ShimmerCommands.wait_for_ack(self)

        # 2.2.7 Send start streaming command
        self.ser.write(struct.pack('B', 0x07))
        ShimmerCommands.wait_for_ack(self)
        print( "start command sending, done.")

        # 2.2.8 Read incoming data and push to LSL stream
        ddata = ""
        numbytes = 0
```

```python
      framesize = 8 # 1byte packet type + 3byte timestamp + 2 byte GSR + 2 byte PPG(Int A13)
      print( "Packet Type\tTimestamp\tGSR\tPPG")
      try:
         while True:
            while numbytes < framesize:
               ddata = self.ser.read(framesize)
               numbytes = len(ddata)

            data = ddata[0:framesize]
            ddata = ddata[framesize:]
            numbytes = len(ddata)

            # read basic packet information
            (packettype) = struct.unpack('B', data[0:1])
            (timestamp0, timestamp1, timestamp2) = struct.unpack('BBB', data[1:4])

            # read packet payload
            (PPG_raw, GSR_raw) = struct.unpack('HH', data[4:framesize])

            # get current GSR range resistor value
            Range = ((GSR_raw >> 14) & 0xff)  # upper two bits
            if(Range == 0):
               Rf = 40.2   # kohm
            elif(Range == 1):
               Rf = 287.0  # kohm
            elif(Range == 2):
               Rf = 1000.0 # kohm
            elif(Range == 3):
               Rf = 3300.0 # kohm

            # convert GSR to kohm value
            gsr_to_volts = (GSR_raw & 0x3fff) * (3.0/4095.0)
            GSR_ohm = Rf/( (gsr_to_volts /0.5) - 1.0)

            # convert PPG to milliVolt value
            PPG_mv = PPG_raw * (3000.0/4095.0)
            timestamp = timestamp0 + timestamp1*256 + timestamp2*65536


            # push data to LSL stream
            gsr_chunk = []
            gsr_chunk.append(GSR_ohm)
            outlet_gsr.push_chunk(gsr_chunk)

```

```
        except: ShimmerCommands.stop_stream(self)
```

Appendix D

EEG_to_LSL.py

```python
# TABLE OF CONTENT
#------------------------------------------------
# 1     Required modules
# 2     GSR_to_LSL
# 2.1   Initialize contact with EEG sensor unit
# 2.2   Define EEG setup
# 2.2.1 Get EEG and AUX channels
# 2.2.2 Define stream info, displayed in LabRecorder
# 2.2.3 Create eeg stream info
# 2.2.4 Create aux stream info
# 2.2.5 Create stream outlet
# 2.2.6 Read incoming data and push to LSL stream


# 1 Required modules
#--------------------
from brainflow.board_shim import BoardShim, BrainFlowInputParams, BoardIds
from pylsl import StreamInfo, StreamOutlet

# 2 EEG_to_LSL
#------------------
class EEG_to_LSL:
    # 2.1 Initialize contact with EEG sensor unit
    def __init__(self, comX):
        BoardShim.enable_dev_board_logger()
        self.params = BrainFlowInputParams()
        self.params.serial_port = comX
        self.EEG_setup()

    # 2.2 Define EEG setup
    def EEG_setup(self):
        board = BoardShim(BoardIds.CYTON_BOARD.value, self.params) # added cyton
        ↪  board id here
        srate = board.get_sampling_rate(BoardIds.CYTON_BOARD.value)
        board.prepare_session()
        board.start_stream()
        #board.config_board('/2')  # enable analog mode only for Cyton Based
        ↪  Boards!    # added from example in docs

        # 2.2.1 Get EEG and AUX channels
        eeg_chan = BoardShim.get_eeg_channels(BoardIds.CYTON_BOARD.value)
        aux_chan = BoardShim.get_accel_channels(BoardIds.CYTON_BOARD.value)
        print('EEG channels:')
        print(eeg_chan)
        print('Accelerometer channels')
        print(aux_chan)

        # 2.2.2 Define stream info, displayed in LabRecorder
        name = 'OpenBCIEEG'
        ID = 'OpenBCIEEG'
        channels = 8
        sample_rate = 250
        datatype = 'float32'
        streamType = 'EEG'
        print(f"Creating LSL stream for EEG. \nName: {name}\nID: {ID}\n")

        # 2.2.3 Create eeg stream info
        info_eeg = StreamInfo(name, streamType, channels, sample_rate, datatype,
        ↪  ID)
        chns = info_eeg.desc().append_child("channels")
        for label in ["AFp1", "AFp2", "C3", "C4", "P7", "P8", "O1", "O2"]:
            ch = chns.append_child("channel")
            ch.append_child_value("label", label)

        # 2.2.4 Create aux stream info
        info_aux = StreamInfo('OpenBCIAUX', 'AUX', 3, 250, 'float32',
        ↪  'OpenBCItestAUX')
        chns = info_aux.desc().append_child("channels")
        for label in ["X", "Y", "Z"]:
            ch = chns.append_child("channel")
            ch.append_child_value("label", label)

        # 2.2.5 Create stream outlets
        outlet_aux = StreamOutlet(info_aux)
        outlet_eeg = StreamOutlet(info_eeg)

        # 2.2.6 Read incoming data and push to LSL stream
        while True:
            data = board.get_board_data() # this gets data continiously

            # don't send empty data
            if len(data[0]) < 1 : continue

            eeg_data = data[eeg_chan]
            aux_data = data[aux_chan]

            # push eeg data to LSL stream
            eegchunk = []
            for i in range(len(eeg_data[0])):
                eegchunk.append((eeg_data[:,i]).tolist())
            outlet_eeg.push_chunk(eegchunk)
            # push aux data to LSL stream
            auxchunk = []
            for i in range(len(aux_data[0])):
                auxchunk.append((aux_data[:,i]).tolist())
            outlet_aux.push_chunk(auxchunk)
```

Appendix E

ShimmerCommands.py

```python
# TABLE OF CONTENT
#-------------------------------------------------------------
# 1. Required modules
# 2. ShimmerCommands
# 2.1 Initialize class
# 2.2 Function: conenct to sensor unit
# 2.3 Function: wait for acknowledge respnse from sensor unit
# 2.4 Function: stop stream of data from sensor unit


# 1. Required modules
#---------------------
import sys, struct, serial

# 2. ShimmerCommands
#--------------------
class ShimmerCommands:

    # 2.1 Initialize class
    def __init__(self):
        self.ser = self.serial_connect()

    # 2.2 Function: conenct to sensor unit
    def serial_connect(self):
        if len(sys.argv) < 2:
            print( "no device specified")
            print( "You need to specify the serial port of the device you wish to connect to")
            print( "example:")
            print( "   aAccel5Hz.py Com12")
            print( "or")
            print( "    aAccel5Hz.py /dev/rfcomm0")
        else:
            self.ser = serial.Serial(sys.argv[1], 115200)
            self.ser.flushInput()
            print( "port opening, done.")
        return self.ser

    # 2.3 Function: wait for acknowledge respnse from sensor unit
    def wait_for_ack(self):
        ddata = ""
        ack = struct.pack('B', 0xff)
        while ddata != ack:
            ddata = self.ser.read(1)
        return

    # 2.4 Function: stop stream of data from sensor unit
    def stop_stream(self):
        KeyboardInterrupt()
        #send stop streaming command
        self.ser.write(struct.pack('B', 0x20))
        print( "stop command sent, waiting for ACK_COMMAND")
        ShimmerCommands.wait_for_ack(self)
        print( "ACK_COMMAND received.")
        #close serial port
        self.ser.close()
        print( "All done")
```

NeuroKit2 analysis

June 10, 2022

1 NeuroKit2 Analysis

The following example of how to extract features from GSR and ECG is copied and replicated from the NeuroKit2. The GSR example can be found here: https://neurokit2.readthedocs.io/en/latest/examples/eda.html, and ECG here: https://neurokit2.readthedocs.io/en/latest/examples/heartbeats.html

1.1 GSR

The galvanic skin response (GSR) can also be refered to as electrodermal activity (EDA), which is what NeuroKit2 calls it.

[1]:
```
# Import necessary packages
import neurokit2 as nk
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
# %matplotlib inline
```

[2]:
```
plt.rcParams['figure.figsize'] = [50, 15]
plt.rcParams['font.size']= 30
```

[3]:
```
gsr_path = '/home/cathrine/Documents/master/project-thesis/src/XDF/csv/
 ↪Shimmer_GSR_kate_output.csv'
gsr_df = pd.read_csv(gsr_path, index_col='timestamp')
gsr_df.head()
```

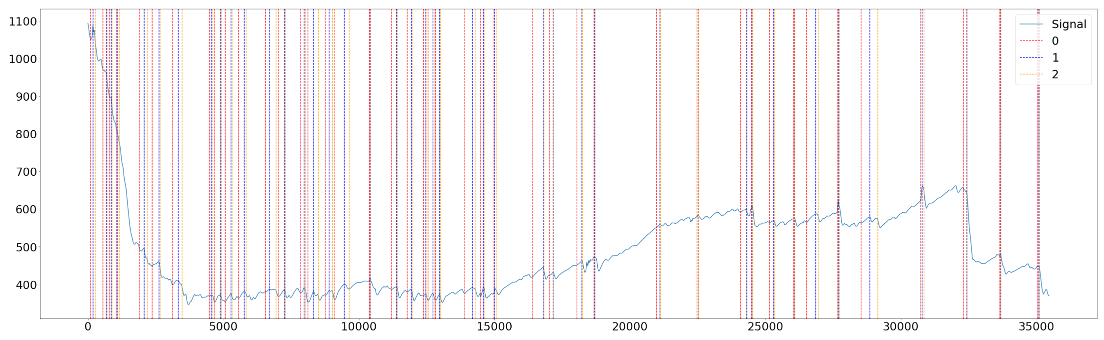
[3]:
```
                    CH1
timestamp
3461.250261  1093.5907
3461.270250  1093.0599
3461.290238  1093.0599
3461.310226  1092.5297
3461.330215  1091.4708
```

[4]:
```
# Process the raw GSR signal
gsr_signal = gsr_df['CH1']
signals, info = nk.eda_process(gsr_signal, sampling_rate=128)
```

Locate Skin Conductance Response (SCR) features, by detecting the location of 1) peak onsets, 2) peak amplitude, and 3) half-recovery time.

```
[5]: # Extract clean EDA and SCR features
     cleaned = signals["EDA_Clean"]
     features = [info["SCR_Onsets"], info["SCR_Peaks"], info["SCR_Recovery"]]
```

```
[6]: # Visualize SCR features in cleaned GSR signal
     plot = nk.events_plot(features, cleaned, color=['red', 'blue', 'orange'])
```

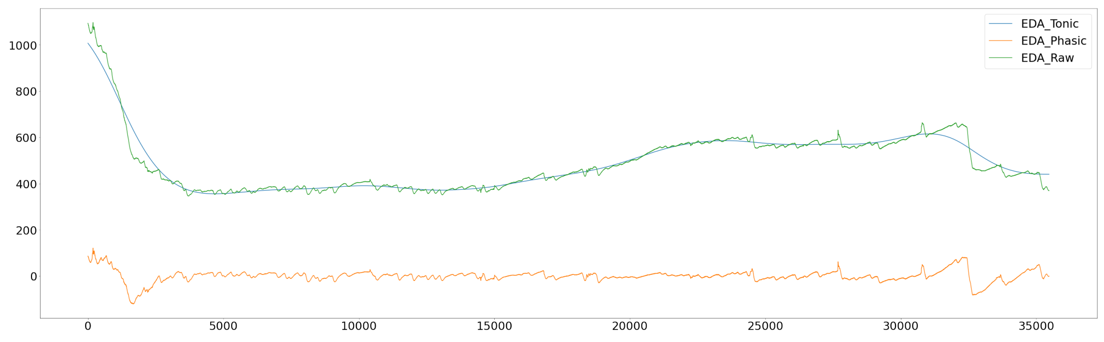


Decompose EDA into Phasic and Tonic components

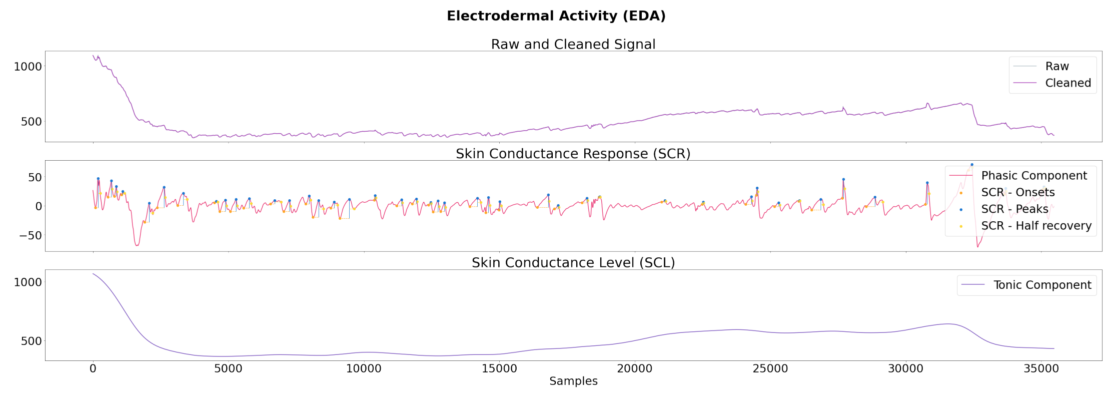
```
[7]: # Filter phasic and tonic components
     data = nk.eda_phasic(gsr_signal, sampling_rate=250)
```

```
[8]: data["EDA_Raw"] = signals["EDA_Raw"]   # Add raw signal
     data.plot()
```

```
[8]: <AxesSubplot:>
```



```
[9]: # Plot all GSR features
     plot = nk.eda_plot(signals)
```



1.2 ECG

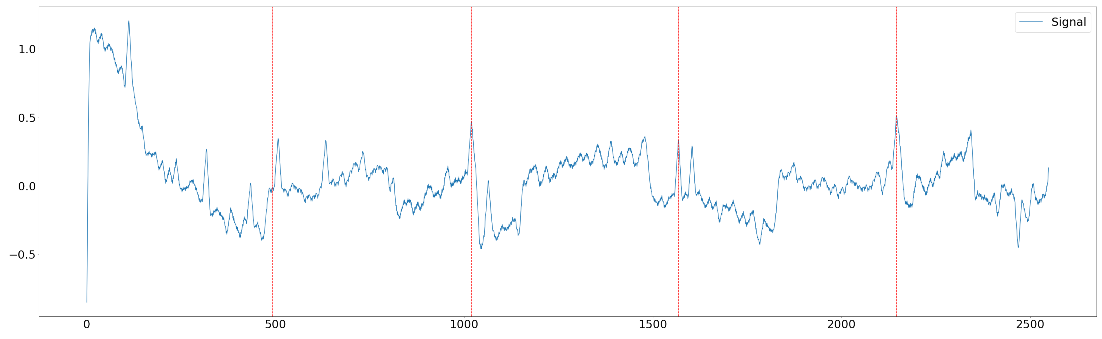
[10]:
```
ecg_path = '/home/cathrine/Documents/master/project-thesis/src/XDF/
 ↪Shimmer_ECG_ecg17.csv'
ecg_df = pd.read_csv(ecg_path, index_col='timestamp', nrows=2550)
ecg_df.head()
```

[10]:
```
                 C1CH1     C1CH2     C2CH1     C2CH2
timestamp
5944.846539  0.003678  0.775452 -1.333962 -2.363281
5944.848058 -2.363281  0.901448  0.945442  0.007212
5944.849578  1.181641  0.935129  0.003534  0.073853
5944.851098 -1.172409  0.003606  0.003606 -2.206345
5944.852617 -0.276947 -0.503770 -2.363281  0.928999
```

[11]:
```
# Automatically process the (raw) ECG signal
signals, info = nk.ecg_process(ecg_df['C2CH2'], sampling_rate=512)
```

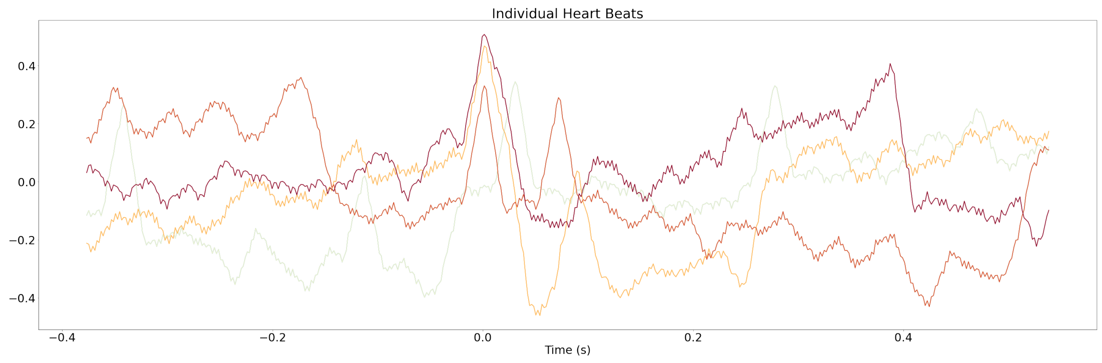
[12]:
```
# Extract clean ECG and R-peaks location
rpeaks = info["ECG_R_Peaks"]
cleaned_ecg = signals["ECG_Clean"]
```

[13]:
```
# Visualize R-peaks in ECG signal
plot = nk.events_plot(rpeaks, cleaned_ecg)
```



Segment the signal around the heart beats

```
[14]: # Plotting all individual heart beats, synchronized by their R peaks
epochs = nk.ecg_segment(cleaned_ecg, rpeaks=None, sampling_rate=512, show=True)
```



MNE EEG analysis

June 8, 2022

1 MNE EEG analysis

The following analysis is copied and replicated from MNE's own example, found here: https://mne.tools/stable/auto_tutorials/preprocessing/30_filtering_resampling.html?highlight=eeg%20drift

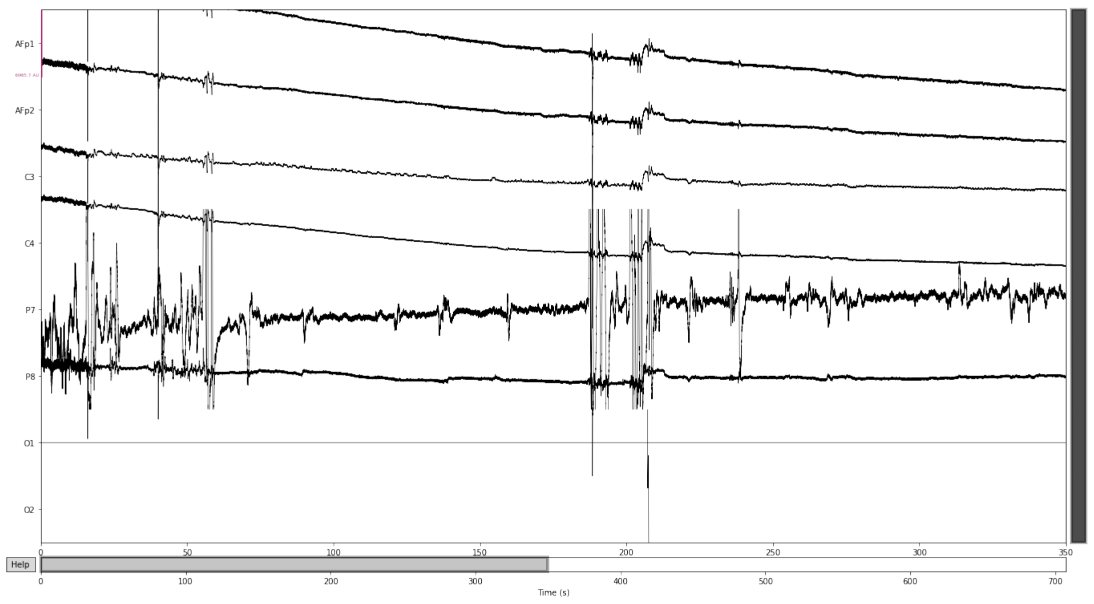
[1]:
```
import numpy as np
import pandas as pd
import mne
import pyxdf
from pyxdf import load_xdf, match_streaminfos, resolve_streams
import matplotlib.pyplot as plt
from mne.datasets import misc
import seaborn as sns
%matplotlib inline
```

[2]:
```
fname = '/home/cathrine/Documents/master/project-thesis/LabRecordings/sub-P001/
 ↪ses-S001/eeg/kate_psychopy_short_successfull_withiut_ecg.xdf'


streams, header = load_xdf(fname)

# Remove all other streams
streams.pop(0)
streams.pop(0)
streams.pop(0)
streams.pop(1)
streams.pop(1)
streams.pop(1)
streams.pop(1)

stream = streams[0]
print(stream['info']['name'][0])


n_chans = int(stream["info"]["channel_count"][0])
fs = float(stream["info"]["nominal_srate"][0])
labels, types, units = [], [], []
try:
    for ch in stream["info"]["desc"][0]["channels"][0]["channel"]:
        labels.append(str(ch["label"][0]))
        if ch["type"]:
            types.append(ch["type"][0])
        if ch["unit"]:
            units.append(ch["unit"][0])
except (TypeError, IndexError):  # no channel labels found
    pass

if not labels:
    labels = [str(n) for n in range(n_chans)]
if not units:
    units = ["NA" for _ in range(n_chans)]

info = mne.create_info(ch_names=labels, sfreq=fs, ch_types="misc")

# convert from microvolts to volts if necessary
scale = np.array([1e-6 if u == "microvolts" else 1 for u in units])

raw = mne.io.RawArray((stream["time_series"] * scale).T, info)


print(raw.info)
```

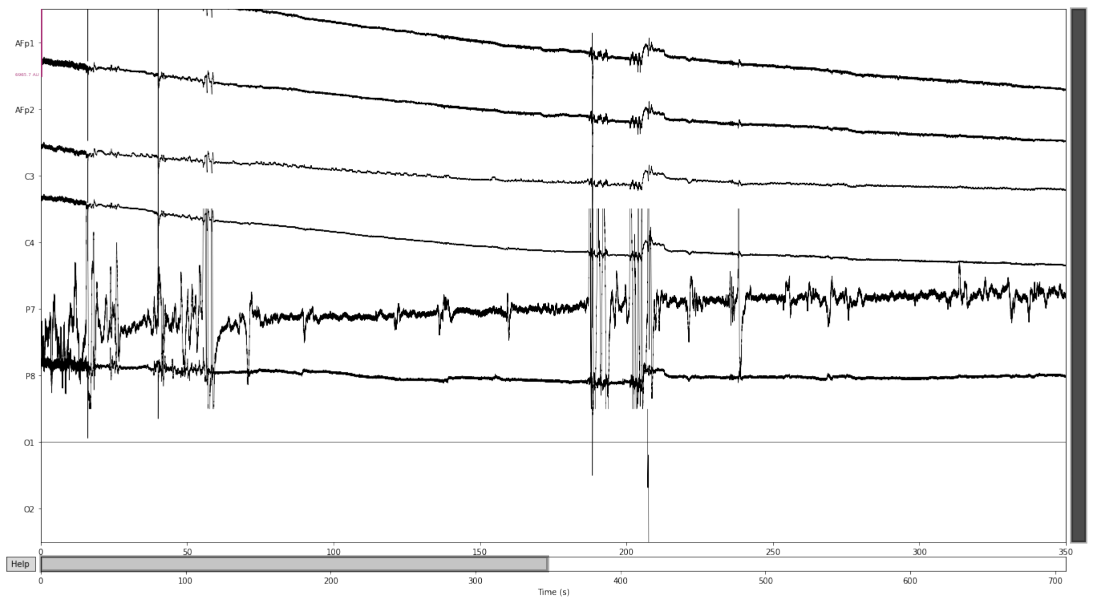
```
OpenBCIEEG
Creating RawArray with float64 data, n_channels=8, n_times=176823
    Range : 0 … 176822 =      0.000 …   707.288 secs
Ready.
<Info | 7 non-empty values
 bads: []
 ch_names: AFp1, AFp2, C3, C4, P7, P8, O1, O2
 chs: 8 misc
 custom_ref_applied: False
 highpass: 0.0 Hz
 lowpass: 125.0 Hz
 meas_date: unspecified
 nchan: 8
 projs: []
 sfreq: 250.0 Hz
>
```

```
[3]: raw.plot(duration=350)
```

```
Using matplotlib as 2D backend.
Opening raw-browser…
```



[3]:

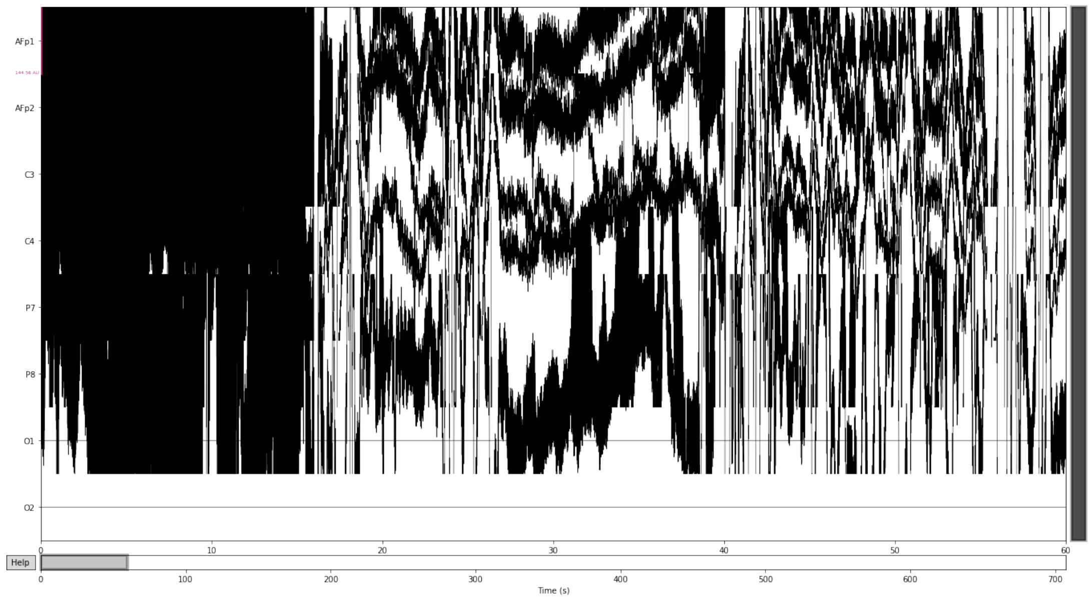


1.1 Slow drifts

Low-frequency drifts in raw data can usually be spotted by plotting a fairly long span of data with the plot() method. The highpass must be set higher then the period of slow drifts. Testing various filters to fully remove slow drifts. Notice that the text output summarizes the relevant characteristics of the filter that was created.

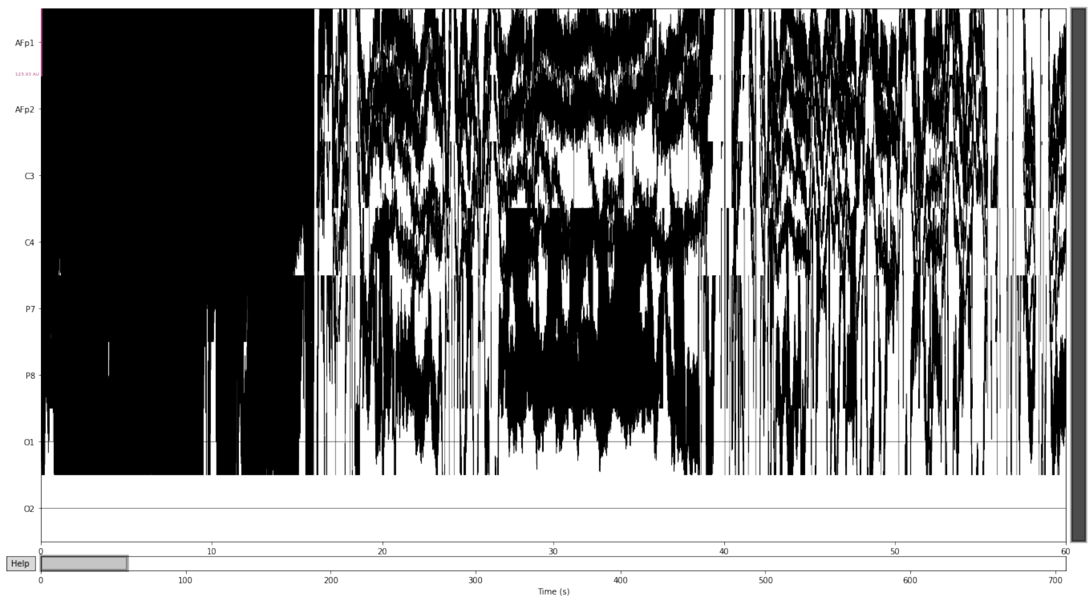
```
[4]: for cutoff in (0.1, 0.5):
         raw_highpass = raw.copy().filter(l_freq=cutoff, h_freq=None, picks='all')
         with mne.viz.use_browser_backend('matplotlib'):
             fig = raw_highpass.plot(duration=60, proj=False,
                                     n_channels=len(raw.ch_names), remove_dc=False)
         fig.subplots_adjust(top=0.9)
         fig.suptitle('High-pass filtered at {} Hz'.format(cutoff), size='xx-large',
                      weight='bold')
```

```
No data channels found. The highpass and lowpass values in the measurement info
will not be updated.
Filtering raw data in 1 contiguous segment
Setting up high-pass filter at 0.1 Hz

FIR filter parameters
---------------------
Designing a one-pass, zero-phase, non-causal highpass filter:
- Windowed time-domain design (firwin) method
- Hamming window with 0.0194 passband ripple and 53 dB stopband attenuation
- Lower passband edge: 0.10
- Lower transition bandwidth: 0.10 Hz (-6 dB cutoff frequency: 0.05 Hz)
- Filter length: 8251 samples (33.004 sec)

Opening raw-browser…
```



```
No data channels found. The highpass and lowpass values in the measurement info
will not be updated.
Filtering raw data in 1 contiguous segment
```

```
Setting up high-pass filter at 0.5 Hz

FIR filter parameters
---------------------
Designing a one-pass, zero-phase, non-causal highpass filter:
- Windowed time-domain design (firwin) method
- Hamming window with 0.0194 passband ripple and 53 dB stopband attenuation
- Lower passband edge: 0.50
- Lower transition bandwidth: 0.50 Hz (-6 dB cutoff frequency: 0.25 Hz)
- Filter length: 1651 samples (6.604 sec)

Opening raw-browser…
```

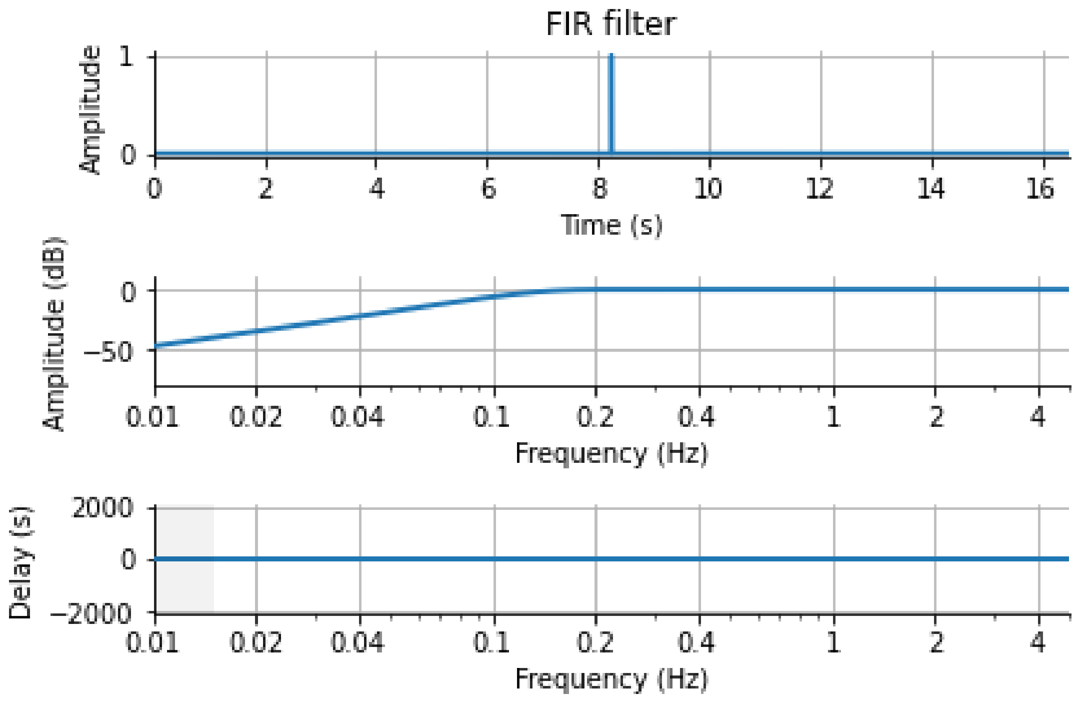


A visualization of the filter:

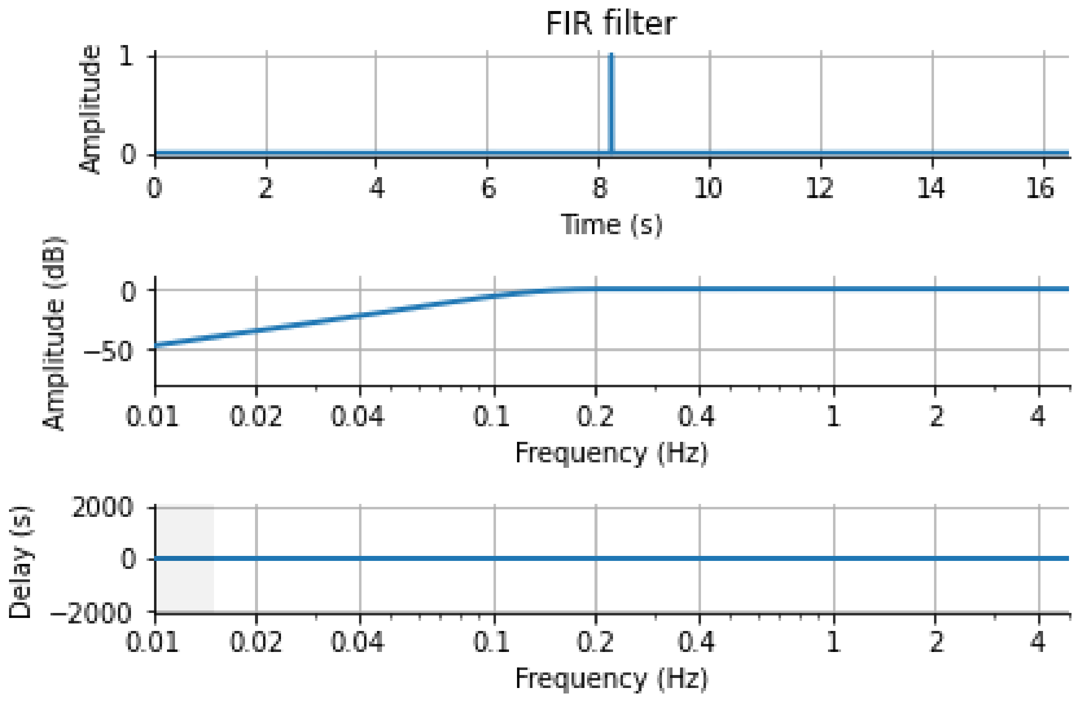
```
[5]: filter_params = mne.filter.create_filter(raw.get_data(), raw.info['sfreq'],
                                              l_freq=0.2, h_freq=None)
```

```
Setting up high-pass filter at 0.2 Hz

FIR filter parameters
---------------------
Designing a one-pass, zero-phase, non-causal highpass filter:
- Windowed time-domain design (firwin) method
- Hamming window with 0.0194 passband ripple and 53 dB stopband attenuation
- Lower passband edge: 0.20
- Lower transition bandwidth: 0.20 Hz (-6 dB cutoff frequency: 0.10 Hz)
- Filter length: 4125 samples (16.500 sec)
```

[6]:
```
mne.viz.plot_filter(filter_params, raw.info['sfreq'], flim=(0.01, 5))
```



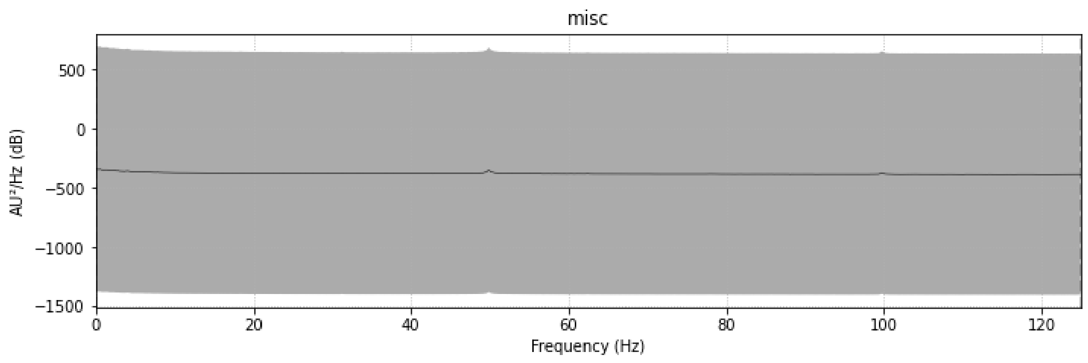
[6]:



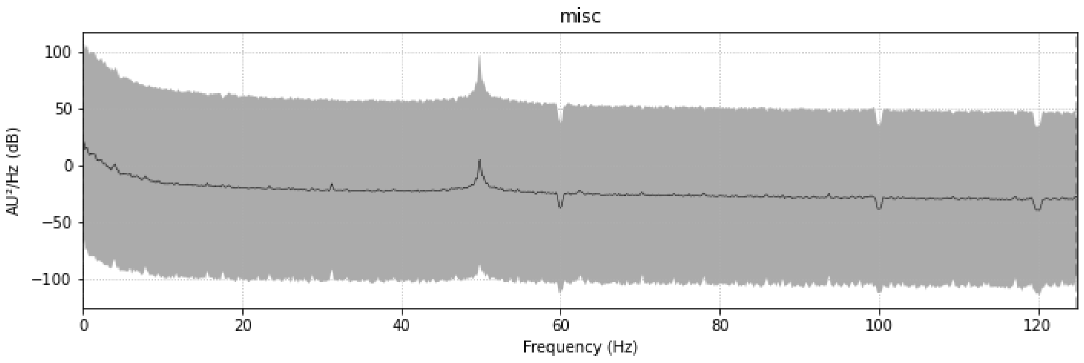
```
[7]: freqs = (60, 100, 120)
     raw_notch = raw.copy().notch_filter(freqs=freqs, picks=['AFp1', 'AFp2', 'C3',␣
      ↪'C4', 'P7', 'P8', 'O1', 'O2'])
     for title, data in zip(['Un', 'Notch '], [raw, raw_notch]):
         fig = data.plot_psd(picks=['AFp1', 'AFp2', 'C3', 'C4', 'P7', 'P8', 'O1',␣
      ↪'O2'], average=True)
         fig.subplots_adjust(top=0.85)
         fig.suptitle('{}filtered'.format(title), size='xx-large', weight='bold')
```

```
Setting up band-stop filter

FIR filter parameters
---------------------
Designing a one-pass, zero-phase, non-causal bandstop filter:
- Windowed time-domain design (firwin) method
- Hamming window with 0.0194 passband ripple and 53 dB stopband attenuation
- Lower transition bandwidth: 0.50 Hz
- Upper transition bandwidth: 0.50 Hz
- Filter length: 1651 samples (6.604 sec)

Effective window size : 8.192 (s)
```

```
/tmp/ipykernel_35444/1244525464.py:4: UserWarning: Infinite value in PSD for
channel O1.
These channels might be dead.
  fig = data.plot_psd(picks=['AFp1', 'AFp2', 'C3', 'C4', 'P7', 'P8', 'O1',
'O2'], average=True)
```



```
Effective window size : 8.192 (s)
```


misc
AU²/Hz (dB)
Frequency (Hz)
100
50
0
−50
−100
0
20
40
60
80
100
120

MNE fNIRS analysis

June 8, 2022

1 MNE fNIRS analysis

The following analysis is copied and replicated from MNE's own example, found here: https://mne.tools/stable/auto_tutorials/preprocessing/70_fnirs_processing.html#sphx-glr-auto-tutorials-preprocessing-70-fnirs-processing-py

```
[1]: import numpy as np
     import pandas as pd
     import mne
     import pyxdf
     import matplotlib.pyplot as plt
     from mne.datasets import misc
     import seaborn as sns
     from mne.preprocessing.nirs import (optical_density,
                                         temporal_derivative_distribution_repair)

     %matplotlib
```

```
Using matplotlib backend: Qt5Agg
```

1.1 Analysis from XDF file of fNIRS data

```
[2]: fname = '/home/cathrine/Documents/master/project-thesis/LabRecordings/sub-P001/
      ↪ses-S001/eeg/kate_psychopy_short_successfull_withiut_ecg.xdf'

     from pyxdf import load_xdf, match_streaminfos, resolve_streams

     streams, header = load_xdf(fname)

     streams.pop(0)
     streams.pop(0)
     streams.pop(0)
     streams.pop(0)
     streams.pop(0)
     streams.pop(0)
     streams.pop(1)

     stream = streams[0]
```

```python
print(stream['info']['name'][0])

n_chans = int(stream["info"]["channel_count"][0])
fs = float(stream["info"]["nominal_srate"][0])
labels, types, units = [], [], []
try:
    for ch in stream["info"]["desc"][0]["channels"][0]["channel"]:
        labels.append(str(ch["label"][0]))
        if ch["type"]:
            types.append(ch["type"][0])
        if ch["unit"]:
            units.append(ch["unit"][0])
except (TypeError, IndexError):  # no channel labels found
    pass


if not labels:
    labels = [str(n) for n in range(n_chans)]
if not units:
    units = ["NA" for _ in range(n_chans)]

info = mne.create_info(ch_names=labels, sfreq=fs, ch_types="fnirs_cw_amplitude")

print('info: ', info)
# convert from microvolts to volts if necessary
scale = np.array([1e-6 if u == "microvolts" else 1 for u in units])

raw_intensity = mne.io.RawArray((stream["time_series"] * scale).T, info)
raw_intensity._filenames = [fname]
first_samp = stream["time_stamps"][0]
```

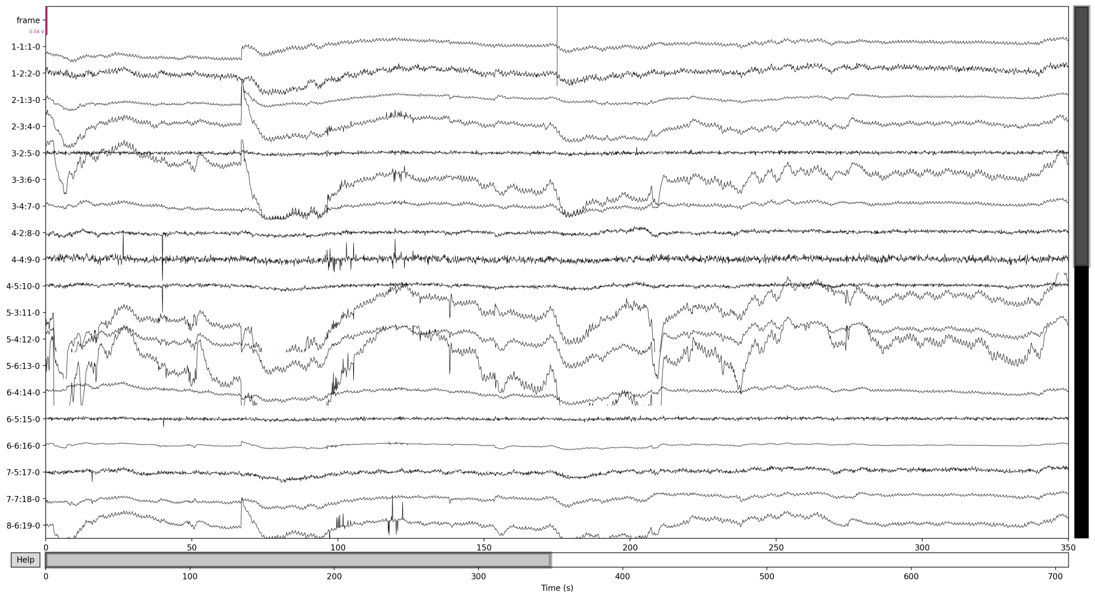
```
NIRStar
info:  <Info | 7 non-empty values
 bads: []
 ch_names: frame, 1-1:1-0, 1-2:2-0, 2-1:3-0, 2-3:4-0, 3-2:5-0, 3-3:6-0, …
 chs: 41 fNIRS (CW amplitude)
 custom_ref_applied: False
 highpass: 0.0 Hz
 lowpass: 3.9 Hz
 meas_date: unspecified
 nchan: 41
 projs: []
 sfreq: 7.8 Hz
>
Creating RawArray with float64 data, n_channels=41, n_times=5540
    Range : 0 … 5539 =      0.000 …   708.992 secs
Ready.
```

```
/tmp/ipykernel_42573/3107810234.py:37: RuntimeWarning: Channel names are not
unique, found duplicates for: {'2-3:4', '8-6:19', '1-2:2', '7-5:17', '4-5:10',
'1-1:1', '2-1:3', '5-4:12', '5-6:13', '8-7:20', '7-7:18', '6-6:16', '4-2:8',
'3-2:5', '5-3:11', '3-4:7', '6-5:15', '4-4:9', '6-4:14', '3-3:6'}. Applying
running numbers for duplicates.
  info = mne.create_info(ch_names=labels, sfreq=fs,
ch_types="fnirs_cw_amplitude")
```

[3]:
```
raw_intensity.plot(duration=350)
```

```
Using matplotlib as 2D backend.
Opening raw-browser…
```

[3]:



1.2 Analysis from NIRX folder of fNIRS data

The data sent from NirStar to LSL is also saved in a folder, containing relevant data. This format is more accepted by the MNE analysis tool, reducing the need for preprocessing, which is required when reading an XDF file.

[4]:
```
fname = '/home/cathrine/Documents/master/project-thesis/LabRecordings/sub-P001/
  ↪ses-S001/fnirs/2022-06-07/2022-06-07_001'

raw_intensity = mne.io.read_raw_nirx(fname, verbose=True)
raw_intensity.load_data()
raw_intensity.plot(duration=350)
```

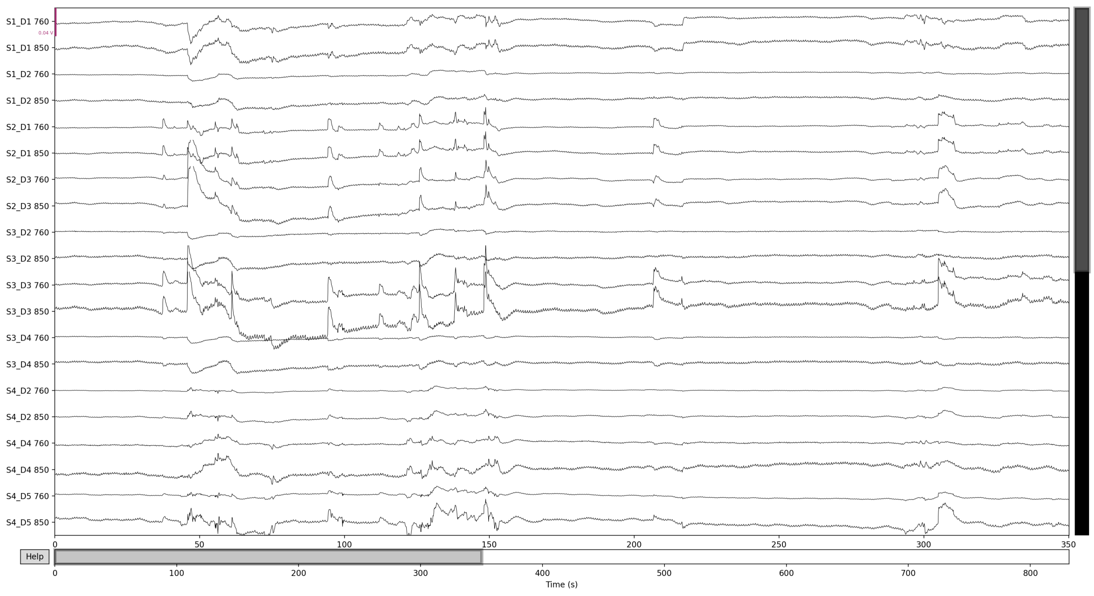
```
Loading /home/cathrine/Documents/master/project-
thesis/LabRecordings/sub-P001/ses-S001/fnirs/2022-06-07/2022-06-07_001
```

```
Reading 0 … 6497  =      0.000 …   831.616 secs…
```

```
/tmp/ipykernel_42573/1017823360.py:3: RuntimeWarning: Extraction of measurement
date from NIRX file failed. This can be caused by files saved in certain locales
(currently only ['en_US.utf8', 'de_DE', 'fr_FR', 'it_IT'] supported). Please
report this as a github issue. The date is being set to January 1st, 2000,
instead of '"tir. 7. jun. 2022""14:00:27.023"'.
  raw_intensity = mne.io.read_raw_nirx(fname, verbose=True)
```

```
Opening raw-browser…
```

[4]:

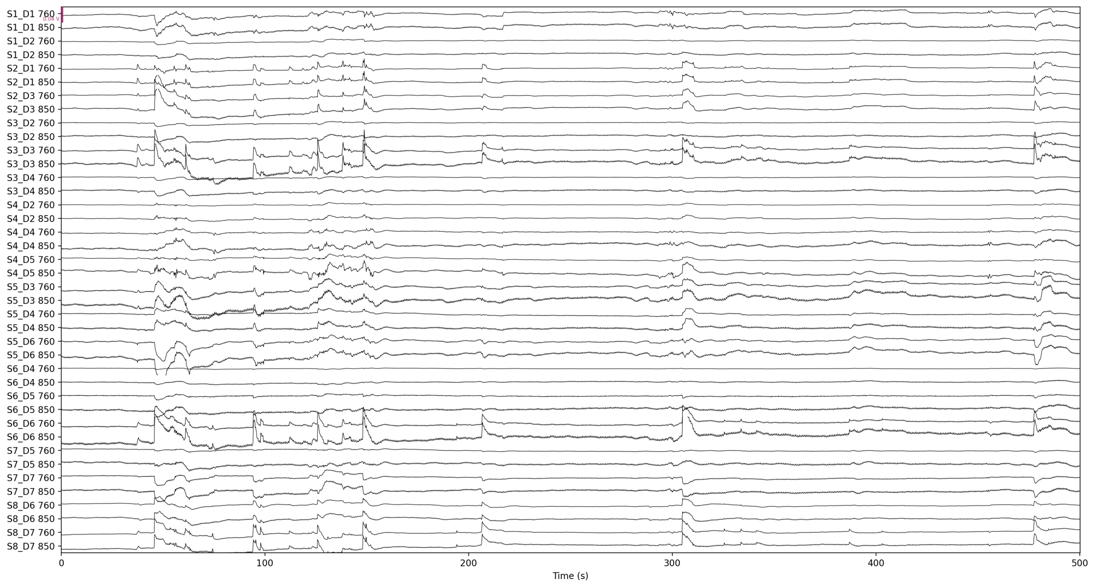


Remove channels that er too close together

[5]:
```python
picks = mne.pick_types(raw_intensity.info, meg=False, fnirs=True)
print(raw_intensity.info)
dists = mne.preprocessing.nirs.source_detector_distances(
    raw_intensity.info, picks=picks)
raw_intensity.pick(picks[dists > 0.01])
raw_intensity.plot(n_channels=len(raw_intensity.ch_names),
                   duration=500, show_scrollbars=False)
```

```
<Info | 9 non-empty values
 bads: []
 ch_names: S1_D1 760, S1_D1 850, S1_D2 760, S1_D2 850, S2_D1 760, S2_D1 …
 chs: 40 fNIRS (CW amplitude)
 custom_ref_applied: False
 dig: 23 items (3 Cardinal, 20 EEG)
 highpass: 0.0 Hz
 lowpass: 3.9 Hz
 meas_date: 2000-01-01 00:00:00 UTC
 nchan: 40
 projs: []
 sfreq: 7.8 Hz
 subject_info: 3 items (dict)
>
Opening raw-browser…
```

[5]:

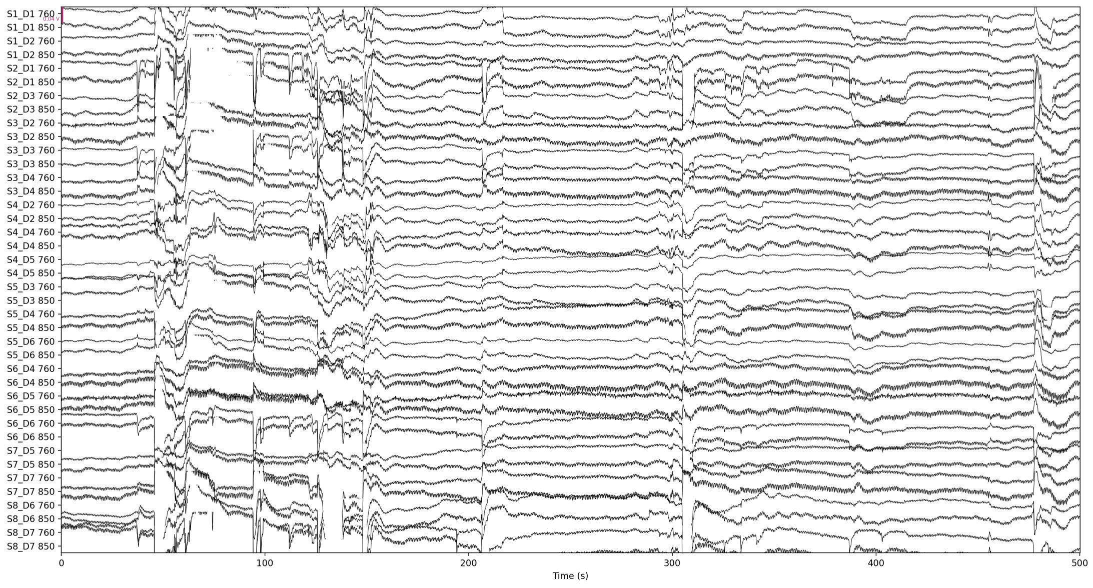


Convert from raw intensity to optical density

[6]:
```python
raw_od = mne.preprocessing.nirs.optical_density(raw_intensity)
raw_od.plot(n_channels=len(raw_od.ch_names),
            duration=500, show_scrollbars=False)
```

```
Opening raw-browser…
```

[6]:

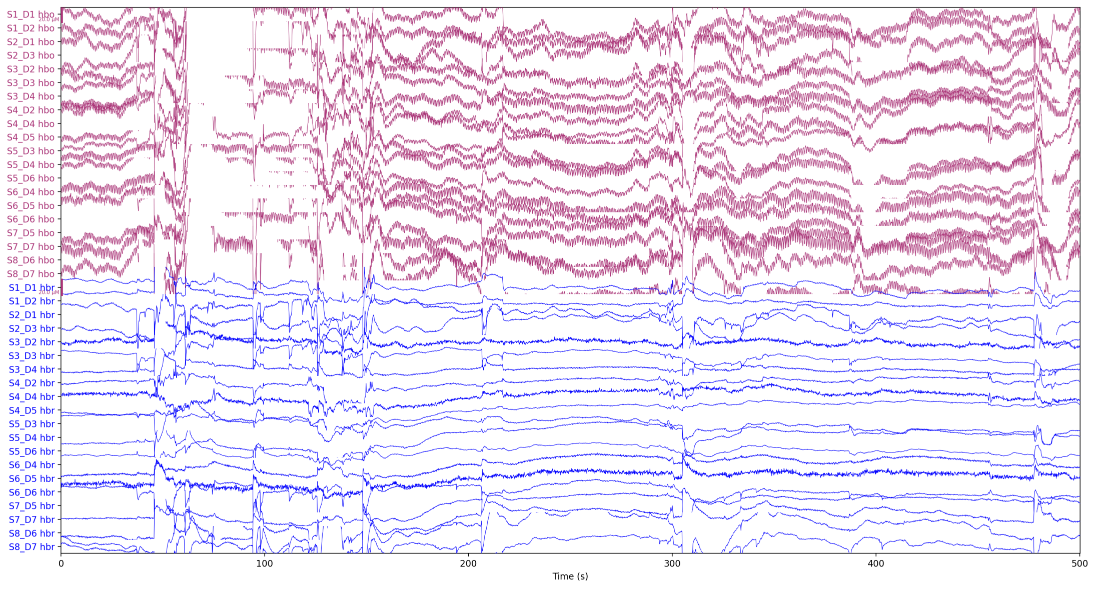


Convert from optical density to haemoglobin

[7]:
```
raw_haemo = mne.preprocessing.nirs.beer_lambert_law(raw_od, ppf=0.1)
raw_haemo.plot(n_channels=len(raw_haemo.ch_names),
               duration=500, show_scrollbars=False)
```

```
Opening raw-browser...
```

[7]:



```
Closing raw-browser...
```