Karlsruher Institut für Technologie

Institut für Telematik
Pervasive Computing Systems / TECO
Leiter: Prof. Dr.-Ing. Michael Beigl

Recognizing Symptoms of Temporomandibular Joint (TMJ) Dysfunction with Wearables

Bachelor's Thesis

by

Alexandr Melnic

Department of Informatics

Responsible Supervisor:	Prof. Dr. Michael Beigl
Supervising Staff:	Tim Schneegans
Project Period:	01.02.2022 - 15.06.2022

Karlsruher Institut für Technologie

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Bachelorarbeit selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel und Quellen benutzt habe, die wörtlich oder inhaltlich übernommenen Stellen als solche kenntlich gemacht und weiterhin die Richtlinien des KIT zur Sicherung guter wissenschaftlicher Praxis beachtet habe.

Karlsruhe, den 15.06.2022

Contents

List of Figures

List of Tables

Abstract

Bruxism is one of the most occurring temporomandibular joint dysfunctions (TMD) characterized by contractions or contraction bursts of masticatory muscles. 80% of bruxers don't even know they have it. If left untreated it can lead to headache, masticatory muscle problems, or teeth fractures. We need cheap, user-friendly, and reliable tools to diagnose and prevent it, as the currently available diagnosis methods tend to be expensive and invasive. In the scope of this study 5 devices were selected and combined together in a custom-built prototype. A user study was designed to simulate bruxism with clenching, grinding, and sliding, as well as comparing it to contrast jaw movements like reading, drinking, chewing, smiling, etc. Traditional machine learning approaches were used in a brute-force approach to classifying bruxism. Four classifiers were mapped with a total of 255 groups of time-series and trained, and an accuracy score was computed for every one of them. This yielded a big sortable table that can be used as a reference for future research. The selection criteria discussed here was to have the highest accuracy while keeping the lowest amount of used time series. We observe that the time series of quaternions from two IMUs placed on the back side of the jaw paired with the random forest classifier, fit these selection criteria best, achieving 88% accuracy in identifying bruxism.

1. Introduction

Cheap and user-friendly wearable devices are needed to diagnose and prevent temporomandibular joint dysfunctions (TMD). Bruxism is one of the most occurring of such dysfunctions. In essence, it is a set of movement patterns of the jaw with different periodicity, force, and duration that occur involuntarily. These patterns include clenching, grinding, and sliding of the jaw. We differentiate between "Awake Bruxism" (AB) and "Sleep Bruxism" (SB) based on the time-period when it occurs. The masticatory muscle contraction and contraction types are also different from person to person and between AB and SB.

This presents a challenge in identifying patients that are experiencing one of the forms of bruxism, but are unaware of it because no damage has been done yet. 80% of bruxism episodes are not accompanied by noise [15]. 80% of bruxers are not aware of their condition [19]. If left untreated it may cause teeth fracture, headache, or masticatory muscle problems.

There isn't a clear consensus about the prevalence of bruxism in the human population, as the numbers are fluctuating around 14 to 20% in children and 16 to 96% in the adult population [11], [19]. The methods for the diagnosis present today are either unreliable or invasive and expensive.

A more convenient and universal way would be the use of one or of a combination of affordable wearable devices. The primary purpose of such wearables can be completely different (e.g. earbuds), but the sensors present on them can be repurposed and trained to classify jaw movements.

Based on previous research in the field, it was interesting to see how different wearables would compare in the same experiment environment. A custom-built prototype was evaluated in a user study. The user study was designed with different motor activities (relevant for AB and SB) in mind.

2. Background

2.0.1 Bruxism

Bruxism represents a series of jaw movements, that can happen at any time of the day and manifest itself as a brief contraction of the masticatory muscles or as multiple contraction bursts in time windows between 0.25 and 2s (table 2.1). The knowledge about motor activity helped design the user study later, as well as determining the sliding window size during the data analysis.

It is also worth mentioning that in the past no strict conclusions were made about the effects of stress on the prevalence of bruxism (e.g. [16]). Still, with the context of the COVID-19 Pandemic, studies showed evidence of bruxism intensification and TMD symptoms caused by the increased stress level [5] [1]. As a result, a long-term stress indicator may also be used as a way of assessing inclination towards bruxism.

2.0.2 Related Work

While researching for previews attempts to classify bruxism, the scope was enlarged to include attempts to classify jaw movements in general.

A considerable amount of effort was invested in the use of wearables with the scope of either classifying AB or using a predefined set of jaw movements (i.e. clenching, teeth tapping) as a way of human-computer interaction (HCI). Electromyography (EMG) is by far not the most popular choice, as it tends to be cumbersome to find the optimal spot to capture the EMG signal of the targeted muscle. Also, the placement of the electrodes on the face is not a practical solution for a daily wearable device.

Table 2.2 captures some highlights used as reference and inspiration for the custom-built prototype and the conducted user study.

Criteria	Classification	Description
When it occurs	AS	Occurs when individual is awake
	SB	Occurs when individual is sleeping
	Combined	Occurs in both situations
Etiology	Primary	No identifiable cause
	Secondary	Secondary to neurologic, psychiatric, sleep or movement disorders, or of an iatrogenic type associated with drug use/withdrawal, etc.
Motor activity type	Tonic	Muscular contractions lasting > 2s
	Phasic	Brief repeated muscular contractions with at least three consecutive electromyographic bursts of 0.25 and 2s duration
	Combined	Variation of tonic and phasic episodes
Activity status	Nonactive	Past bruxism
	Active	Current or present bruxism

Table 2.1: Classification of bruxism [20]

Device	Description & Results
eSense earables	76% on grinding and 73% on clenching detection accuracy [3] 94% on chewing detection accuracy [9] 1% to 4% error rate on chewing detection accuracy [9]
EEG	F1-scores on grinding up to 0.9 using around-the-ear EEG electrodes [8]
Custom earphones	87.6% detection accuracy for face-related movements using barometer placed in earphones [2]
Custom earphones	Reliable detection of up to 7 jaw gestures using earphone speakers as microphones and recording vibrations propagated to the ear-canal [13]
Custom ear-pieces	Recognition of 13 teeth tapping gestures with 90.9% accuracy using custom-built ear-pieces with IMUs on each side of the user's jaw [18]
EMG	100% detection accuracy of bruxism using EMG signals. [17]

Table 2.2: Related Work

3. Implementation

3.1 Hardware

Based on previous research a set of devices and sensors was selected (table 3.1). The selection was done with a goal in mind: to utilize the maximum of the *available space* on a human head and neck. Assuming that those surfaces will allow the recording of the vibrations and movements generated by the jaw bones and muscles.

A Raspberry Pi 3 B+ (RPI) was used as the central processing unit (fig. 3.1). It has a Bluetooth antenna as well as a variety of wired connectivity options to ease the build of this prototype. As discussed later in sec. 3.2 the RPI was used not only for the data collection but as well for the real-time data visualization. The RPI had some hardware limitations to read the raw data directly from some of the sensors. It was solved by using external microcontrollers (see 3.1.1.2, 3.1.1.3 and 3.1.1.4).

All of the cables and attached devices were labeled accordingly to the side they have to be attached to, to maintain a reproducible setup between the participants of the user study.

3.1.1 Peripherals

3.1.1.1 eSense earables

The eSense earables are a pair of true wireless earphones. The data we are interested in is the 6 DOF IMU located only in the left earpiece only. Because of this limitation, two left earpieces were used. Even though the left and right earpieces are not interchangeable, the participants in the user study didn't notice any difference in the fit in the right ear.

The default frequency of sending of the IMU data is set to 10Hz. In the case the audio playback is not being used, the frequency can be increased to up to 100Hz by changing the advertising and connection intervals ([6]), and still, have enough battery life for multiple user study recordings on a single battery charge.

As a result, the IMU data from both earables was polled at 100Hz.

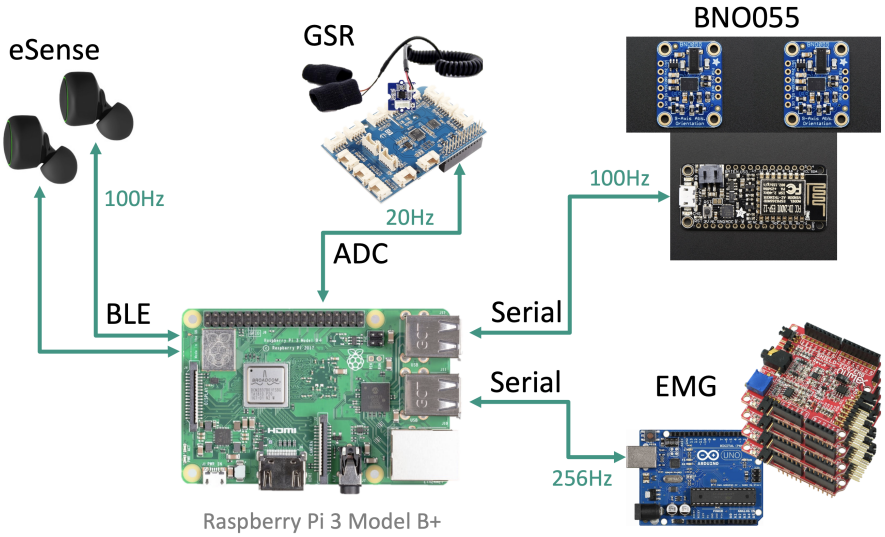


Figure 3.1: Prototype Overview

Device	Amount	Sensor	Area of contact
eSense earbuds 3.1.1.1	2	6 DOF IMU	In ear
Arduino BNO055 3.1.1.2	2	9 DOF IMU	Back side of the jaw (secured with skin-friendly tape)
Olimex EMG-Shield	4	Electromyography (EMG)	Masseter and temporalis muscles
Throat microphone	1	Vibration sensor	Neck
Grove GSR	1	Galvanic skin response (GSR)	Index and middle finger

Table 3.1: Hardware peripherals

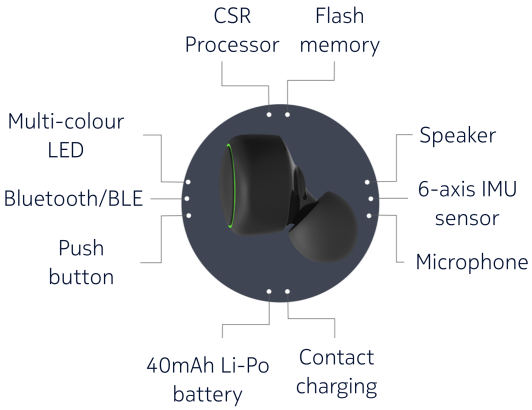


Figure 3.2: eSense earable [12, 7]

The connection was done directly with the RPI by using a python package called `bleak`. The incoming payload is a binary string that was decoded accordingly with the eSense specification [6].

The IMU was kept at the default ranges of $\pm 4g$ for the accelerometer and $\pm 500 deg/s$ for the gyroscope. No further transformations to other units were done.

Interface	BLE
Sample rate	100Hz
Data	Accelerometer & gyroscope

Table 3.2: eSense earable. Used technical specifications

3.1.1.2 Arduino BNO055

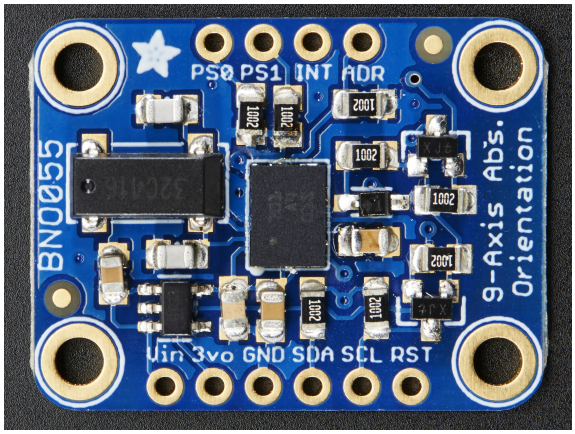


Figure 3.3: Arduino BNO055 breakout board

This (fig. 3.3) small breakout board from Arduino has a BNO055, a 9 DOF IMU with sensor fusion on the chip. The fastest of the available protocols for data transmission on this board is I2C. Fortunately the BNO055 has an `ADR` pin, which set to high will change its I2C address (from `0x28` to `0x29`). This allows communication with both of the used BNO055s over the same I2C bus.

Even though the RPI has pins for the I2C communication, it doesn't support clock stretching (a I2C feature used by the BNO055). Using the RPI the data collection happened at a much lower rate (around 30Hz) and generally, the connection was unreliable.

After testing different microcontrollers, the decision was made to use the Adafruit Feather HUZZAH ESP8266 board (fig. 3.4), to read the raw data from the BNO055s and then send it directly to the RPI. Because the RPI couldn't handle reliably more than 2 BLE connections at the same time (which were already *occupied* by the eSense earables), the IMU data was transmitted over the Serial Protocol (USB).

The serial payload was a string of comma separated IMU values from both BNO055, terminated with a new-line symbol: gyroscope (`x, y, z` from the `x28`), accelerometer (`x, y, z` from the `x28`), quaternion (`w, x, y, z` from the `x28`), gyroscope (`x, y, z` from the `x29`), accelerometer (`x, y, z` from the `x29`), quaternion (`w, x, y, z` from the `x29`).

When operating in the 9 DOF mode, the BNO055 needs to be calibrated every time it's powered on. To automatize this process, the BNO055s were each calibrated once, and the calibration data was stored on the Feather. Every time they were powered on, the Feather will load the calibration data first, before making any readings (see ch. 3.2.1.1).

Interface	Serial
Sample rate	100Hz
Data	Accelerometer & gyroscope & quaternion

Table 3.3: Arduino BNO055 in combination with Feather Huzzah. Used technical specifications

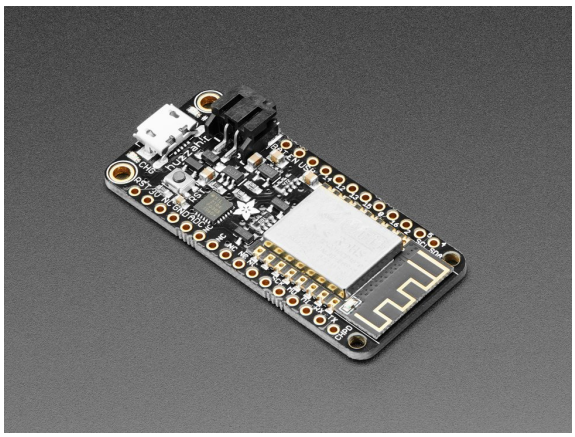


Figure 3.4: Adafruit Feather Huzzah ESP8266 board

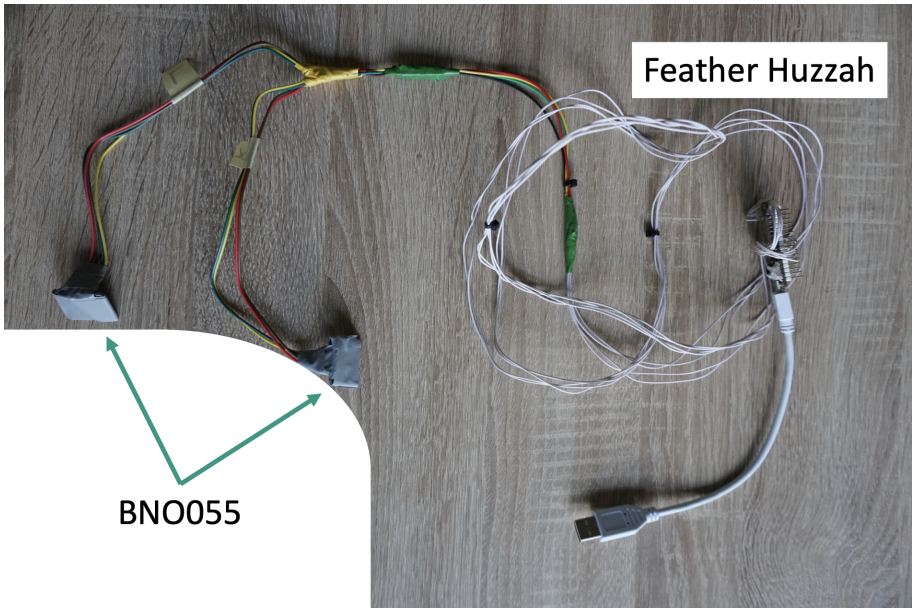


Figure 3.5: Prototype of Arduino BNO055 with Feather Huzzah

3.1.1.3 Olimex EMG-Shield

The EMG-Boards from Olimex can be stacked on top of each other, to allow readings from up to 6 channels at the same time. To capture the EMG signals from the left and right masseter muscles as well as from the left and right temporalis muscles, 4 EMG-Shields were used. The raw data from the board is analog, so an analog-to-digital (ADC) converter was needed because the RPI doesn't have any analog input pins. For this purpose, the Arduino Uno (fig. 3.7) was used. The collected data was sent directly to the RPI over the Serial Protocol (USB).

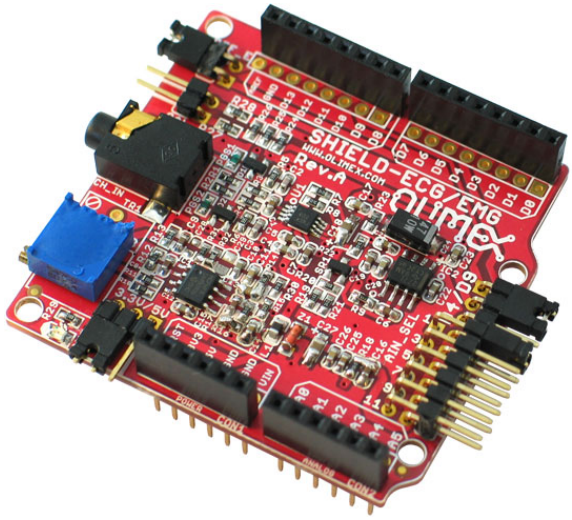


Figure 3.6: Olimex EMG-Shield

The serial payload consisted of 4 integer values ranging from `0` to `1023` terminated with a new-line symbol: EMG signal of the left masseter, right masseter, left temporalis, right temporalis.

Interface	Serial
Sample rate	256Hz
Data	EMG

Table 3.4: Olimex EMG-Shield in combination with Arduino Uno. Used technical specifications

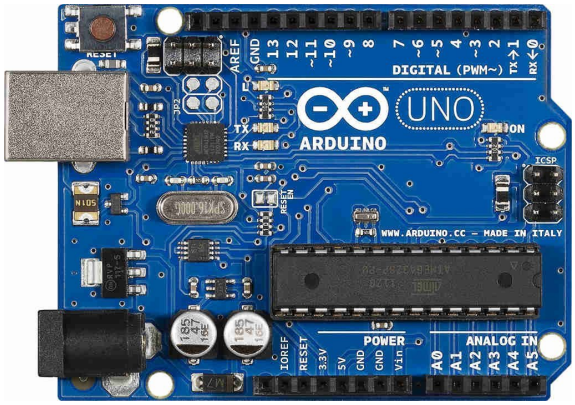
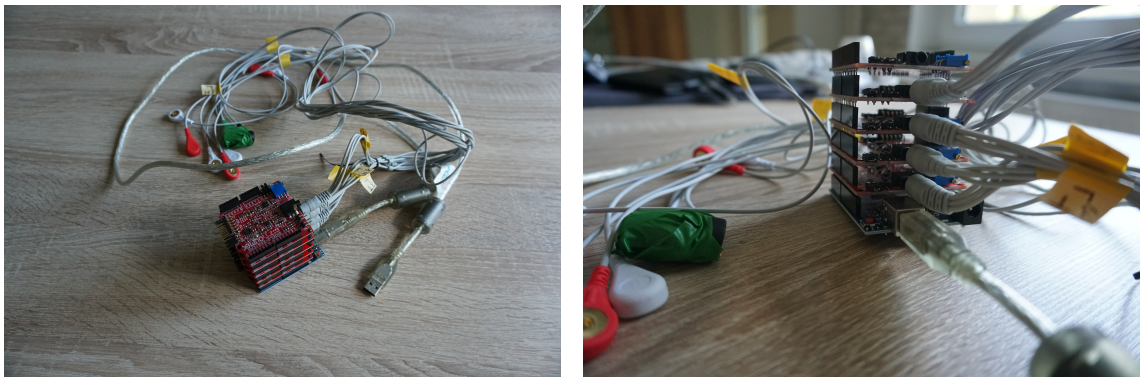


Figure 3.7: Arduino Uno board

3.1.1.4 Throat Microphone

As a novelty in using wearables for bruxism detection, a throat microphone (fig. 3.9) was introduced to the set of peripherals. The AUX present on the RPI doesn't have the functionality to record audio. The microphone was connected directly to a laptop, which was also used to control the RPI. The raw audio data was captured with the Audacity software [4].



(a) Top view (b) Side view

Figure 3.8: Prototype of EMG-Shields with Arduino Uno

Figure 3.9: Throat Microphone

Interface	AUX TRRS
Sample rate	44.1kHz
Data	Raw Audio

Table 3.5: Throat Microphone. Used technical specifications

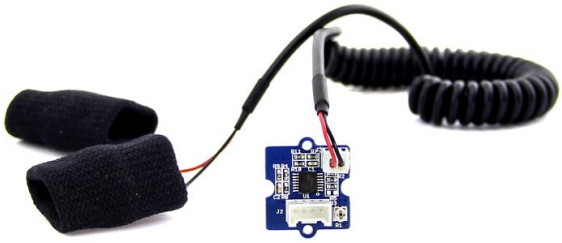


Figure 3.10: Grove GSR

3.1.1.5 Grove GSR

With the Grove GSR, we can measure the electrical conductance of the skin. The raw data of this sensor is just an integer. The higher the value, the more stressed the person appears to be.

Interface	Digital
Sample rate	20Hz
Data	Galvanic skin response

Table 3.6: Grove GSR. Used technical specifications

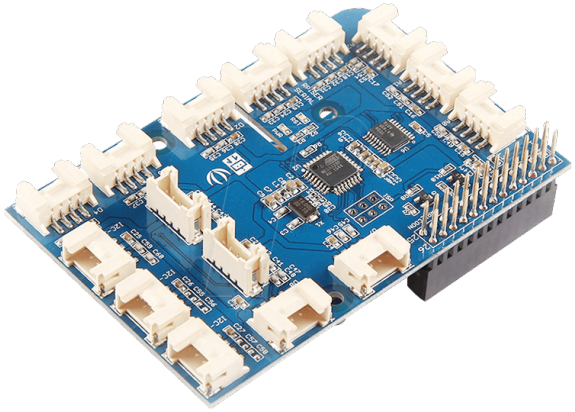
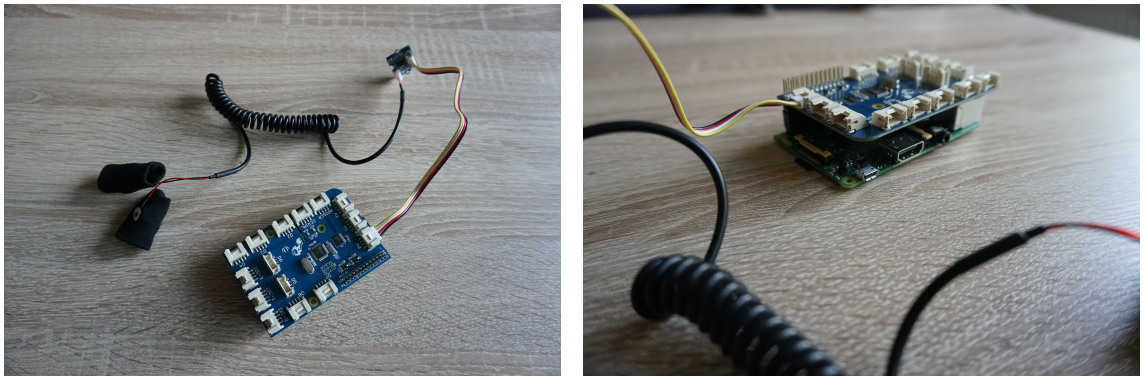


Figure 3.11: Raspberry Pi Shield — Grove Pi Plus



(a) Top view (b) Side view

Figure 3.12: Prototype of Grove GSR with Grove Pi Plus

3.2 Software

To record and visualize the raw data required a custom framework with the following criteria:

- **Fast**, to avoid any data loss;
- **Flexible**, it should be easy to add new devices, no matter what the underlying data transmission protocol is being used;

- **Reliable**, the captured data should be timestamped at the capture time, and be saved as soon as possible to persistent storage (par. 3.2.2.1).

To sample the data at the desired rate, first, we had to ensure that the peripherals are able to send the data at *that* sample rate.

The data collection framework is implemented to be indifferent to the underlying sensor type and data transmission protocol. The yielded data was saved in a standardized format. A generic real-time data visualizer was additionally implemented.

3.2.1 Microcontrollers

3.2.1.1 Feather Huzzah

As described in ch. 3.1.1.2, the communication between both BNO055s and the Feather was implemented via the I2C protocol. During every user study, the BNO055 with the address `0x28` was always on the *right* back side of the jaw, and the BNO055 `0x29` on the *left* side accordingly.

```
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BNO055.h>
#include <utility/imumaths.h>

//                                   id, address
Adafruit_BNO055 bno0 = Adafruit_BNO055(-1, 0x28);
Adafruit_BNO055 bno1 = Adafruit_BNO055(-1, 0x29);

adafruit_bno055_offsets_t bno_offset28 = { -39, -18, -39, 123, 211, -400, -1, -1,
↪  0, 1000, 747 };
adafruit_bno055_offsets_t bno_offset29 = { -12, -28, -46, 25, 437, -66, -2, 1, 0,
↪  1000, 718 };

sensors_event_t orientationData[2], angVelocityData[2], linearAccelData[2],
↪  magnetometerData[2], accelerometerData[2], gravityData[2];
imu::Quaternion quat[2];
```

Listing 1: BBO055.ino init

On each power cycle, the devices are initialized and the calibration data is uploaded. Additionally, an external crystal (the one present on the breakout instead of the one present on the chip) is used. This increases the overall accuracy of the recorded data.

The data is then collected sequentially and sent over serial with a baudrate of 500,000.

No delays were implemented in the main loop, as it isn't possible to read the raw data from the sensors faster than the designed 100Hz.

```
void setup(void)
{
  Serial.begin(500000);

  /* Initialise the sensor */
  setupBNO(&bno0, &bno_offset28);
  setupBNO(&bno1, &bno_offset29);
}

void setupBNO(Adafruit_BNO055* bno, adafruit_bno055_offsets_t* offsets) {
  if (!bno->begin())
  {
    /* There was a problem detecting the BNO055 ... check your connections */
    Serial.print("Ooops, no BNO055 detected ... Check your wiring or I2C ADDR!");
    while (1);
  }
  delay(1000);

  bno->setSensorOffsets(*offsets);
  delay(1000);

  bno->setExtCrystalUse(true);
}
```

Listing 2: BBO055.ino setup

```
void loop(void)
{
  readData(&bno0, 0);
  readData(&bno1, 1);

  printData(0);
  Serial.print(",");
  printData(1);

  Serial.println();
}
```

Listing 3: BBO055.ino main loop

3.2.1.2 Arduino Uno

Each of the EMG-Shields has a jumper to select the analog channel the raw data should be sent. A mapping of the analog pin to muscle was made and kept consistent through the user studies table 3.7.

The sample rate of the EMG signals is soft-locked at 256Hz. This is the optimal operational frequency for this shield from Olimex.

Analog pin	Muscle
1	Left masseter
2	Right masseter
3	Left temporalis
4	Right temporalis

Table 3.7: EMG analog pin to muscle mapping

```
// All definitions
#define NUMCHANNELS 4
#define SAMPFREQ 256                        // ADC sampling rate 256
#define PERIOD_us (1000000/(SAMPFREQ))      // Set 256Hz sampling frequency

// Global constants and variables
unsigned char CurrentCh;          //Current channel being sampled.
unsigned long last_us = 0L;       // Helper for the sample rate
```

Listing 4: EMG.ino init

The analog pins are being read sequentially and sent over serial with a baudrate of 115,200.

```
void setup() {
 Serial.begin(115200);
}

void loop() {
  if (micros() - last_us > PERIOD_us) {
    last_us += PERIOD_us;

    //Read the 4 ADC inputs
    for(CurrentCh = 0; CurrentCh < NUMCHANNELS; CurrentCh++){
        Serial.print(analogRead(CurrentCh));

      if (CurrentCh != NUMCHANNELS - 1) {
        Serial.print(",");
      }
    }
    Serial.println();
  }
}
```

Listing 5: EMG.ino setup and main loop

3.2.2 Raspberry Pi

A data collection framework (3.2.2.1) and a real-time data visualizer (3.2.2.2) were implemented for the RPI. Both can be deployed on any system capable of running python version 3.7. The only restriction would be the set of peripherals and the availability of the required communication protocols on the said systems. Which can still be easily adjusted if necessary.

3.2.2.1 Data Collection Framework

Overview

The data collection framework is a command-line application written in python. It provides a generic interface to add new sensors, as well as the ability to create virtual sensors with a predefined sample rate. It accepts the name of the participant as the single command-line argument.

As it runs completely on the RPI, an Ethernet cable was used to connect it over SSH. Before each user study, the script is started with a code-name generated by the participant. From this point the activity of the script can be divided into two phases:

1. Sequentially, initialize every in-code defined sensor; Spawn 2 main coroutines per defined sensor (see par. 3.2.2.1); Spawn keyboard event listener coroutine.
2. Run spawned coroutines concurrently until a keyboard event listener coroutine will be triggered to stop the script execution.

The concurrency is achieved with the `asyncio` package. To be able to achieve the desired speed in data collection, all of the I/O-related tasks (as sensor readings) should be written in a non-blocking fashion. All coroutines are prepended with the `async` keyword in the signature. To signal a coroutine switch the `await` keyword is used before the call to a coroutine.

```
1 if __name__ == "__main__":
2     logging.basicConfig(format='%(asctime)s - %(message)s',
3                         datefmt='%d.%m.%Y %H:%M:%S', level=logging.INFO)
4
5     dir_name = sys.argv[1]
6     halt_event = asyncio.Event()
7     asyncio.run(main(dir_name, halt_event))
```

Listing 6: Data collection framework entry point

Sensor Interface

The generic sensor interface implements the reactor pattern (list. 8): An asynchronous producer and an asynchronous consumer is initialized for every sensor. The purpose of the producer is either

```python
async def main(dir_name, halt_event):
    SENSORS = [
        # Sensor(name="mock-s0"), # --- virtual sensor definition
        BLE_eSense(name="ble-left", ble_device_name="eSense-0091"),
        BLE_eSense(name="ble-right", ble_device_name="eSense-0398"),
        GSR_Grovepi(name="gsr"),
        EMG_Olimex_x4(name="emg"),
        BNO055_x2(name="bno"),
        # ...
    ]

    device = Bruxi(dir_name=dir_name, sensors=SENSORS, halt_event=halt_event)

    logger.info(
        f"Initialising {len(device.sensors)}/ {len(SENSORS)} sensors..")
    await device.initialize_sensors()
    logger.info("Done.")

    cli_event_handler = ainput(halt_event)

    logger.info("Spawning workers..")
    await asyncio.gather(*device.spawn_coroutines(), cli_event_handler)
    logger.info("All workers exited.")
```

Listing 7: Data collection framework main event loop

- to poll the data at the predefined sample rate (i.e. the Feather with BNO055s, see 3.1.1.2 and 3.2.1.1),
- or to react to incoming notifications from an external device (i.e. BLE notifications from the eSense earable, see 3.1.1.1).

The sensor producer and consumer share the same data buffer, an asynchronous queue. As the data is incoming, it is labeled with the current timestamp, and transformed in the desired `dict` structure. This payload is asynchronously put to the left side of the queue. The consumer on the other hand has an always running `while` loop, which asynchronously checks the shared queue for new payloads. If the queue is not empty, and no sensor has its data ready, the payload is removed from the right side of the queue and saved to the corresponding `.csv` file (list. 9).

This sensor structure is the key to having a fast and reliable data collection framework.

Data structure

The sensor data for every participant is saved in multiple `.csv` files in a folder named with the participant-provided code-name. The `.csv` files are named after the name of the sensor origin. The sensor names, as well as the headers for the resulted `.csv` files, are defined in-code.

3.2.2.2 Data Visualizer

The sensors implement different communication protocols and there was no standard way of visualizing the data. Also, the visualization of multiple sensors at the same time in one single place was not possible. Using the standard output of the data collection framework (3.2.2.1), it was straightforward to connect a data visualizer, to use the development of the prototype.

Overview

The real-time data visualizer consists of two main components:

1. Python websocket server;
2. Vue web application.

Python Websocket Server

Using the structure described in 3.2.2.1, the websocket sever can stream the list of the code-names in the output directory on the RPI. Every such directory is a collection of `.csv` files, that can be streamed to the connected client (i.e. web browser on an external device). The most recent 100 lines from each `.csv` file are streamed at a 60Hz frequency.

Vue web application

The web application is built with the Vue framework. Once connected to the websocket, the available folders will be displayed as a list of buttons. Upon clicking on a button the data stream will start (fig. 3.13).

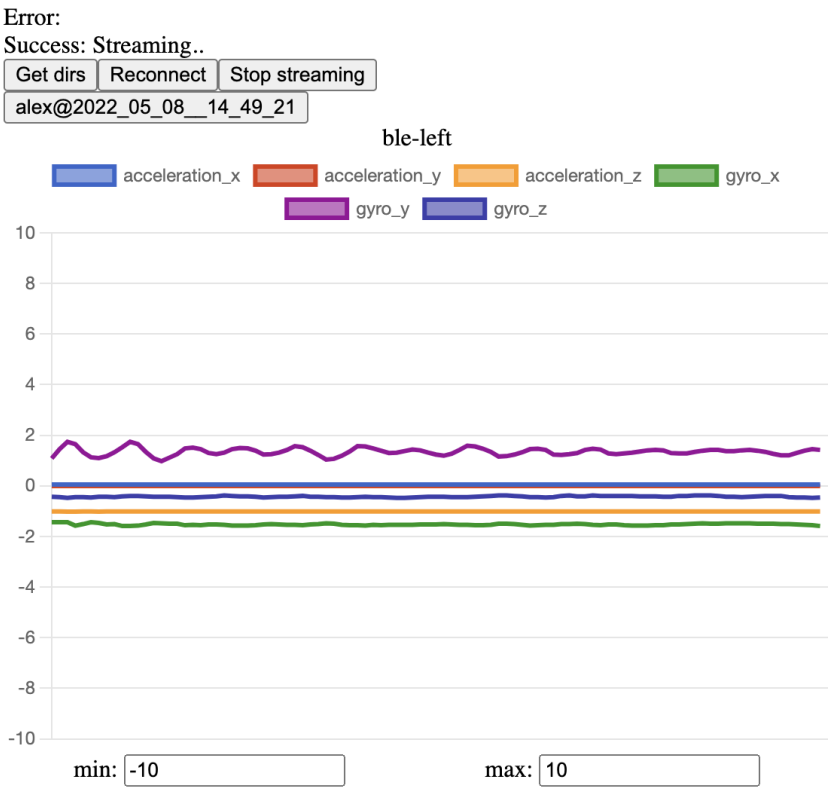


Figure 3.13: Vue web application example

```python
class Sensor:
    """Interface to ease the data collection"""

    def __init__(self, name: str, sample_rate_s: float = 0.01, csv_headers:
    ↪  List[str] = ["dt", "column0"]) -> None:
        self.name = name
        self.sample_rate_s = sample_rate_s
        self.producer = Producer()
        self.csv_headers = csv_headers
        self.consumer = Consumer(filename=f"{self.name}.csv",
                queue=self.producer.queue,
                finished_execution=self.producer.finished_execution,
                csv_headers=csv_headers)

    async def _init(self) -> None:
        """If there are methods to be awaited for the sensor initialization, add
        ↪  them here.
        Later you can call this method in the main loop
        """
        pass

    async def get_data(self) -> dict:
        """Override this method with proper pyshical sensor read
        """
        return {
            "dt": now(),
            "column0": 42.0
        }

    async def start_stream(self) -> None:
        """Start streaming sensor data
        Should call `self.queue(data)`
        """
        while not self.producer.finished_execution.is_set():
            # Simulate N-Hz sample rate
            if self.sample_rate_s > 0:
                await sleep(self.sample_rate_s)

            data = await self.get_data()

            await self.queue(data)

    async def queue(self, data) -> None:
        """Abstraction to produce data with the producer
        """
        await self.producer.produce(data=data)
```

Listing 8: Generic Sensor interface

```python
class Consumer:

    async def consume(self) -> None:
        with open(f"{self.dir}/{self.filename}", 'a') as f:
            writer = csv.writer(f)
            while not self.finished_execution.is_set():
                # wait for an item from the producer
                item = await self.queue.get()

                # append a new row to the .csv file
                writer.writerow(item.values())

                # remove the payload from the queue
                self.queue.task_done()
```

Listing 9: Consumer' consume method

4. Evaluation

The scope of the user study was to collect data from participants simulating bruxism as well as performing contrast jaw movements. All of the sensors but the throat microphone were recorded with the RPI. The throat microphone was connected directly to the laptop used to control the RPI.

A total of 10 participants, 1 female, were recruited, with the youngest participant aged 19 and the oldest aged 28. After inspecting the data, 4 recordings of the temporalis muscles were removed from the resulting dataset (3 recordings of the left temporalis muscle, 1 recording of the right temporalis muscle). In one of the cases, the EMG electrodes disconnected themselves during the study. In the other cases, the data was very noisy. 2 audio recordings were also discarded. The challenges during the data collection are discussed in ch. 4.2.

The experiment took about 30min. The collected usable data had a length of about 16min. Throughout the experiment, the participants were asked to stand still if not asked otherwise, especially in the phase with the bruxism-related tasks, and to focus their eyes on a circle on the wall in front of them. At every step, they were instructed by the experimenter about the exercise they had to perform and the moment they had to start and stop performing it. An overview of the user study can be seen in figure 4.1.

4.1 Data collection

A total of 8 raw audio files were collected with the throat microphone. Six complete sets of time-series (a full set having N = 37 time-series) were used. In one set of time-series, the EMG recordings of the right temporalis muscle were corrupted (leaving n = 36 time-series). In the other three sets of time-series, the EMG recordings of the left temporalis muscle were corrupted (leaving n = 36 time-series for each set). The study yielded a total of 366 time-series collected with the RPI and eight audio recordings, with a total data removal of 1% for the RPI (representing 10% data removal of the collected EMG time-series) and 20% for the audio recordings.

The time-series had a length of around 16min. The 37 time-series consists of the x, y, and z axis of the left (right) eSense earables gyroscope, x, y, and z axis of the left

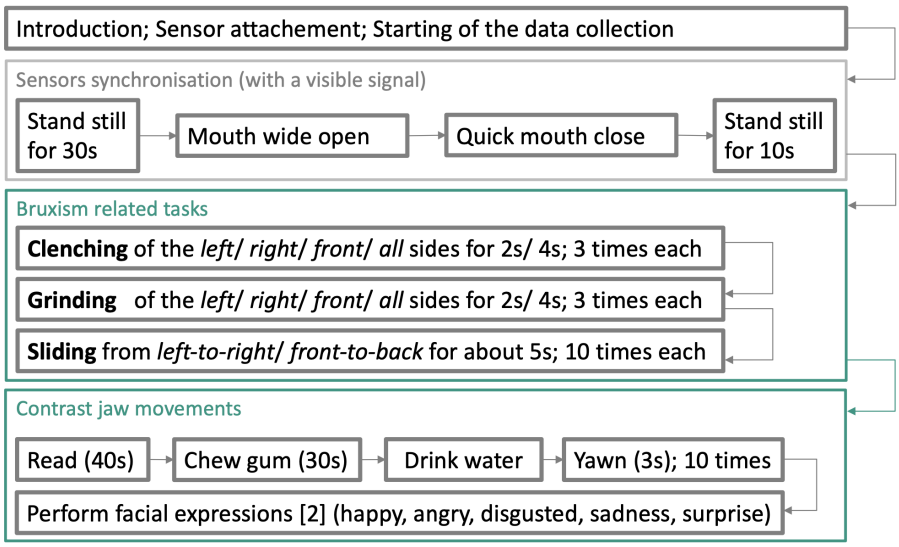


Figure 4.1: User study overview

(right) eSense earables accelerometer, x, y, and z axis of the left (right) BNO055 gyroscope, x, y and z axis of the left (right) BNO055 accelerometer, w, x, y and z axis of the left (right) BNO055 quaternion, EMG signal for the left (right) masseter, left (right) temporalis, and GSR (table 4.1).

Device	Time-series	Sample Rate
eSense earables	Left gyroscope `x, y, z` Left accelerometer `x, y, z` Right gyroscope `x, y, z` Right accelerometer `x, y, z`	100Hz 100Hz
BNO055	Left gyroscope `x, y, z` Left accelerometer `x, y, z` Left quaternion `w, x, y, z` Right gyroscope `x, y, z` Right accelerometer `x, y, z` Right quaternion `w, x, y, z`	100Hz 100Hz
EMG	Left masseter Right masseter Left temporalis Right temporalis	256Hz
Throat Microphone	Raw audio	44.1kHz
GSR	GSR	20Hz
Total	37 time-series & 1 audio file	

Table 4.1: Collected data overview

4.2 Sensor Attachment

The sensor attachment was done as described in the table 3.1. Some of the peripherals required special mounting based on the participants anatomy (see 6.3.2) and

some of the participants had trouble wearing some types of peripherals (see 6.3.1, 6.3.3).

4.3 Study Design

The participants were introduced to the user study and the concept of bruxism. After generating a pseudo-random code-name and acknowledging and accepting the terms of the data collection the sensors were attached and the data collection started.

4.3.1 Sensor synchronization

Even though the RPI already synchronizes the collected data from the connected sensors, using its internal clock, the throat microphone was connected to a separate device. To be able to align these time-series later, the participants were asked to perform a certain jaw movement, as a synchronization signal:

1. Stand still for 30s;
2. Mouth wide open;
3. Quick mouth close;
4. Stand still for 10s.

4.3.2 Bruxism related tasks

In this phase the participants were asked to perform the following jaw movements to simulate bruxism:

- **Clenching**: touch the requested sides with as much force as possible, without other jaw movements.
- **Grinding**: touch the requested sides with as much force as possible, with additional small jaw movements. Imagine that you are rubbing the touching teeth with one another;
- **Sliding**: move your lower jaw in the requested pattern.

The sequence of performing of these movements was repeated several times with different duration:

1. Clench the *left* side for **2s**, 3 times in total, with pauses in-between;
2. Clench the *right* side for **2s**, 3 times in total, with pauses in-between;
3. Clench the *front* side for **2s**, 3 times in total, with pauses in-between;
4. Clench *all sides* for **2s**, 3 times in total, with pauses in-between;
5. pause for 5s.
6. Clench the *left* side for **4s**, 3 times in total, with pauses in-between;

7. Clench the *right* side for **4s**, 3 times in total, with pauses in-between;
8. Clench the *front* side for **4s**, 3 times in total, with pauses in-between;
9. Clench *all sides* for **4s**, 3 times in total, with pauses in-between;

Similarly grinding was done next.

Then sliding was performed with the movement of the whole jaw:

1. Slide from *left-to-right* for about **5s**, 10 times in total;
2. Slide from *front-to-back* for about **5s**, 10 times in total;

4.3.3 Contrast jaw movements

After the bruxism phase, it was important to record other typical jaw movements. For that following materials were prepared:

- A text snippet (in both English and German language), with a time-to-read of about 40s;
- Chewing gum;
- A glass of water;
- 6 facial expressions from the KDEF [10], printed on paper.

The tasks were then done in the following order:

1. Read the following text out loud, either in English or German;
2. Chew a gum for **30s**;
3. Drink a glass of water;
4. Yawn for about **3s** or longer if you feel so, 10 times in total;
5. Perform the following facial expression as you see it for about **4s**, relax your face, and proceed with the next facial expression, 12 times in total.

After that, the data collection was stopped. The raw audio data was saved on the laptop and the generated `.csv` files were copied from the RPI to the laptop using `scp`. The whole user study duration was about 35min.

4.4 Survey

Each participant completed a survey at the end of the user study. This included demographic questions and questions about the wear and feel of the peripherals. The results are presented in the table 4.2, 4.3 and 4.4. It appears that the participants found the BNO055s the most comfortable.

	Std. dev.	Var.	Min	Max
Visible difference between masseter muscles	1.12	1.06	-2	2

Table 4.2: Masseter muscle asymmetry (-2 = left side is bigger, 2 = right side is bigger)

	Std. dev.	Var.	Min	Max
Throat microphone	1.07	1.03	0	3
eSense earables	0.77	0.88	0	3
EMG electrodes	0.28	0.53	2	3
BNO055	0.28	0.53	2	3

Table 4.3: Participants opinion on the devices comfort (0 = very uncomfortable, 3 = very comfortable).

Device	Std. dev.	Var.	Min	Max
Throat microphone	0.12	0.02	0.70	0.97
eSense earables	0.10	0.01	0.71	1
EMG electrodes	0.17	0.03	0.43	1
BNO055	0.10	0.01	0.75	1

Table 4.4: Device score based on participants' answers on overall wear and feel (0 = lowest score, 1 = highest score).

4.5 Analysis

4.5.1 Labeling

Data irregularity

The sensors are set to deliver data at a predefined sample rate. But in practice, this value fluctuates somewhat. As a result, the datapoints of the collected time-series are not aligned. Even the data of a single time-series may be captured at different intervals while maintaining the designed sample rate. In such cases, the data we deal with is *irregular* (e.g. 4.2).

To remove the irregularity from the data and make it *regular*, the time-series can be resampled (e.g. 4.3). At this moment, we can choose the resampling rate equal to the original sampling rate (i.e., aligning the datapoints of a time-series to fixed intervals) or select a higher or lower sample rate. As we don't require high-precision timestamps in the nanoseconds range, it is enough to have precision in the tens or hundreds of milliseconds. After some trials, the datasets were resampled to 20Hz. The missing values were linearly interpolated.

Label methodology

The labeling was done manually after the user study concluded. To ease this process, the following procedure was done:

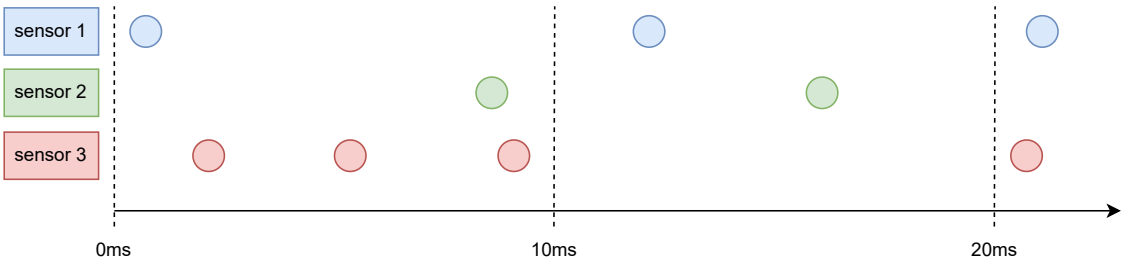


Figure 4.2: Example of irregular data. Circles represent the moment a payload was captured. Colors relate the point in time to the sensor of origin

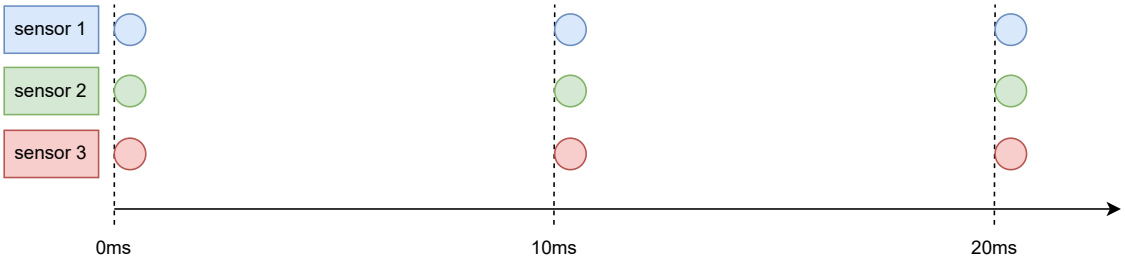


Figure 4.3: Example of regular data. Circles represent the moment a payload was captured. Colors relate the point in time to the sensor of origin

1. For each dataset, the maximum of the minimum timestamps was calculated (i.e., the first timestamp of the last sensor that started delivering data). Then all timestamps in a dataset were offset by the calculated timestamp. As a result, if plotted, having the time on the X-axis, the time-series will start at 0.
2. The values for each time-series were z-normalized, and the plots were presented overlapping in a single figure (one of such datasets is shown in fig. A.8).

To z-normalize the values for each time-series the following formula was used:

$$y := \frac{y - \mu}{\sigma}$$

where y is a value in the time-series, μ is the mean, and σ is the standard deviation.

The data aligned perfectly, and different activity types performed during the user study can be precisely located.

Label types

All the time ranges with the bruxism-related tasks were labeled as 1 (i.e., is bruxism). The rest of the datapoints else was labeled as 0 (i.e., no bruxism).

Window selection

A sliding window approach was applied, which is the standard procedure in time-series data processing. The sliding window had a length of 1.6s with 50% overlap

(0.8s). This ensures that we are grouping enough data points together and not missing transitions between muscle contractions or contraction bursts. The window was labeled by the dominant event in that window; If the larger portion of the window contains bruxism data and a smaller portion of silent data, the window would be labeled as bruxism. If the split is equal, then the window is labeled as silent [3].

This resulted in windows having 32 datapoints for each of the 37 time-series, yielding 1184 features.

4.5.2 Classification

The classification was done by using a brute-force approach. Four traditional machine learning classifiers from sklearn were trained: random forest (RF), k-nearest neighbors (kNN), support vector machine (SVM), and decision tree (DT).

Brute-force Approach Overview

The time-series were generalized into the following groups:

```
timeseries_groups = {
    "bno_gyro": ['x28_gyro_x', 'x28_gyro_y', 'x28_gyro_z', "x29_gyro_x",
    ↪  "x29_gyro_y", "x29_gyro_z"],
    "bno_acc": ["x28_acceleration_x", "x28_acceleration_y", "x28_acceleration_z",
    ↪  "x29_acceleration_x", "x29_acceleration_y", "x29_acceleration_z"],
    "bno_quat": ["x28_quaternion_w", "x28_quaternion_x", "x28_quaternion_y",
    ↪  "x28_quaternion_z", "x29_quaternion_w", "x29_quaternion_x",
    ↪  "x29_quaternion_y", "x29_quaternion_z"],

    "ble_gyro": ["gyro_x", "gyro_y", "gyro_z", "gyro_x_2", "gyro_y_2",
    ↪  "gyro_z_2"],
    "ble_acc": ["acceleration_x", "acceleration_y", "acceleration_z",
    ↪  "acceleration_x_2", "acceleration_y_2", "acceleration_z_2"],

    "gsr": ["gsr"],

    "masseter": ["masseter_left", "masseter_right"],

    "temporalis": ["temporalis_left", "temporalis_right"]
}
```

Listing 10: Dataset time-series abstractions

then a power set (excluding the empty set) was computed for the set of the groups of time-series: `"bno_gyro"`, `"bno_acc"`, `"bno_quat"`, `"ble_gyro"`, `"ble_acc"`, `"gsr"`, `"masseter"`, `"temporalis"` (8 in total). For every two-sided sensor, both sides were placed in the same group. Bruxers tend to have one side of the masticatory muscles overdeveloped, which will lead to cleaner data samplings and better results.

But this is highly individual, and to cover every possible use-case scenario, the data from both sides is needed.

The resulted power set had a length of 255 ($2^8 - 1$)

For Every such group, an array of windows with the corresponding time-series from every dataset was computed. The window labels from every dataset were stored in an array as well so that a window will share the same positional index in the array as its label does.

Brute-force training

Every feature set (group of time-series) windows were split in 20:80 ratio (20 for testing, 80 for training) and passed sequentially to a model classifier.

Brute-force evaluation

For every model—feature-set pair, an accuracy score was calculated.

5. Results

5.1 Overview

The top performing models—feature-set pair was RF (`"bno_gyro", "bno_quat", "ble_acc", "masseter", "temporalis"`) with a score of `0.915`.

Table 5.1 depicts the top 5 best performing models sorted by the feature-set size. It was sorted this way because, alongside the theoretical highest score, it is also important that the feature-set has a practical application in real life. As an example RF (`"bno_quat"`) (score = 0.881) will be valued higher ten the RF (`"bno_gyro", "bno_quat", "ble_acc", "masseter", "temporalis"`) (score = 0.915). It's mainly because it's much easier to design a less error-prone prototype using just one type of sensor. Also, it is a much more practical and non-invasive solution for the end-user to wear.

Clearly, RF outperforms other classifiers by far. But the most considerable insight here is that the `bno_quat` appears in every top for every feature-set size. It gives two main takeaways:

- The sensor fusion (the process of calculating the quaternion using the 9 DOF data provided by the IMU) of the BNO055 helps to increase the IMU accuracy for bruxism detection;
- A fine-tuned EMG may ultimately be the ground truth in bruxism assessment (achieving 100% accuracy [17]). Still, a pair of IMUs remains cheaper and more practical, and this study also showed a good enough option for bruxism detection.

5.2 Data Quality

eSense earables & BNO055

The IMU time-series were generally clean and didn't present any challenge or corruption (e.g. A.1).

EMG

The EMG quality signal is very dependent on the placement of the electrodes. It can produce precise and indistinguishable patterns (e.g., fig. A.2, A.3, A.5, A.4). Otherwise, if the electrodes were misplaced even by 1mm, the time-series had to be discarded, as nothing other than noise was recorded (e.g. fig. A.6).

Throat Microphone

The throat microphone performed poorly, and it wasn't able to catch the vibrations from the bruxism. The audio files were not used later in the classification.

GSR

The collected data showed a similar pattern among all participants: Higher values in the bruxism simulation phase, followed by an immediate decay during the contrast jaw movements phase (fig. A.7).

Model, feature-set pair	Score	Feature-set size
RF (bno_quat)	0.881	1
kNN (bno_quat)	0.880	1
DT (bno_quat)	0.849	1
RF (ble_acc)	0.846	1
RF (bno_acc)	0.844	1
RF (bno_quat,temporalis)	0.904	2
RF (bno_quat,ble_gyro)	0.896	2
RF (bno_gyro,bno_quat)	0.890	2
kNN (bno_quat,ble_acc)	0.888	2
RF (bno_quat,gsr)	0.887	2
RF (bno_quat,ble_gyro,temporalis)	0.906	3
RF (bno_gyro,bno_acc,bno_quat)	0.904	3
RF (bno_gyro,bno_quat,gsr)	0.897	3
RF (bno_acc,bno_quat,ble_acc)	0.894	3
RF (bno_gyro,bno_quat,masseter)	0.894	3
RF (bno_gyro,bno_quat,ble_acc,temporalis)	0.907	4
RF (bno_quat,ble_gyro,gsr,temporalis)	0.906	4
RF (bno_gyro,bno_acc,ble_acc,temporalis)	0.905	4
RF (bno_gyro,bno_quat,gsr,temporalis)	0.905	4
RF (bno_gyro,bno_quat,ble_gyro,temporalis)	0.903	4
RF (bno_gyro,bno_quat,ble_acc,masseter,temporalis)	0.915	5
RF (bno_acc,bno_quat,ble_acc,gsr,temporalis)	0.913	5
RF (bno_gyro,bno_quat,ble_gyro,masseter,temporalis)	0.913	5
RF (bno_gyro,bno_acc,bno_quat,masseter,temporalis)	0.910	5
RF (bno_gyro,bno_acc,bno_quat,ble_acc,temporalis)	0.909	5
RF (bno_gyro,bno_acc,bno_quat,ble_gyro,masseter,temporalis)	0.915	6
RF (bno_gyro,bno_acc,bno_quat,ble_gyro,gsr,temporalis)	0.913	6
RF (bno_gyro,bno_acc,bno_quat,ble_gyro,ble_acc,temporalis)	0.911	6
RF (bno_gyro,bno_quat,ble_acc,gsr,masseter,temporalis)	0.910	6
RF (bno_gyro,bno_quat,ble_gyro,ble_acc,gsr,temporalis)	0.905	6
RF (bno_acc,bno_quat,ble_gyro,ble_acc, gsr,masseter,temporalis)	0.913	7
RF (bno_gyro,bno_acc,bno_quat,ble_gyro, ble_acc,masseter,temporalis)	0.906	7
RF (bno_gyro,bno_acc,bno_quat,ble_gyro, ble_acc,gsr,temporalis)	0.902	7
RF (bno_gyro,bno_acc,bno_quat,ble_gyro, ble_acc,gsr,masseter)	0.900	7
RF (bno_gyro,bno_acc,bno_quat,ble_gyro, gsr,masseter,temporalis)	0.900	7
RF (bno_gyro,bno_acc,bno_quat,ble_gyro,ble_acc, gsr,masseter,temporalis)	0.901	8
DT (bno_gyro,bno_acc,bno_quat,ble_gyro,ble_acc, gsr,masseter,temporalis)	0.852	8
kNN (bno_gyro,bno_acc,bno_quat,ble_gyro,ble_acc, gsr,masseter,temporalis)	0.701	8
SVM (bno_gyro,bno_acc,bno_quat,ble_gyro,ble_acc, gsr,masseter,temporalis)	0.670	8

Table 5.1: Model, feature-sets pairs with score. Only top 5 values are shown for each feature-set size

.

6. Conclusions

6.1 Summary

A brute-force approach was used to classify bruxism-related events. Four traditional classifiers were combined with different recorded time-series sets. Quaternion data from the IMUs placed on the back sides of the jaw with the random forest classifier yielded the best performance score with the least amount of sensors. It achieved 88% accuracy. The best performing score overall was 91.5% achieved by the random forest classifier with the BNO055 gyroscope and quaternion, eSense earables accelerometer, masseter, and temporalis time-series (from both sides each). As the IMUs tend to be cheaper than the EMG and are already present in various wearables (e.g. eSense earables) it once more confirms their potential in bruxism detection.

This study summarizes the use of 5 different devices with the scope of classifying bruxism-related events. The COVID-19 Pandemic may have also started another wave of a bruxism pandemic. The overall increase in the stress level may as well increase the prevalence of awake bruxism. A cheap and practical device is needed, that will have the ability to diagnose potential patients before it will lead to discomfort or teeth damage. The device selection was done based on previous research in the field and then combined together in a custom-built prototype. While selecting the devices we tried to maximize the use of the *potential area* on the human head and neck, where a bruxism-induced signal may be recorded. A user study was designed to include bruxism-related jaw movements (clenching, grinding, and sliding) and contrasting *regular* jaw movements (talking, drinking, chewing, smiling, etc.). All of the selected devices for the prototype were used at the same time during every user study.

6.2 Limitations & Future Work

The study was done in a controlled environment, and the presented results may only be compared only in the context of this study. It is important to mention, that the custom-built prototype was just used as a framework to generate a reference.

The results of this study show that the sensor fusion of the IMU data appears the most promising feature to have a wearable device in the vicinity of the jaw. The closer to the jaw, the better the accuracy.

Bruxism happens mostly unconsciously. Prolonged periods appear during a patient's sleep. An EMG-oriented device may actually be a better choice for a sleep use case. A comparison with a much more mature device with IMUs is needed to be done in-the-wild.

Alternative ways of labeling and data segmentation can be explored. In an in-the-wild environment, bigger duration trends can be tracked. The performance in such cases should ideally correlate with the lessening of the bruxism occurrence over time.

More advanced machine learning approaches for time-series classification may also be useful.

The labeling of the data was in itself a very cumbersome and error-prone process. As we had to rely on visual clues in the time-series, as well as on timing (estimated duration of performed tasks and the duration of the pauses in-between). The keyboard event listener coroutine (see ch. 3) can be improved and extended to create the labels during the runtime. Either the experimenter or the participant itself can press (or hold) a button, to indicate a start or end of an action. This coroutine can be completely extracted from the framework and used as a separate device or as a software module.

6.3 Challenges

6.3.1 eSense earables

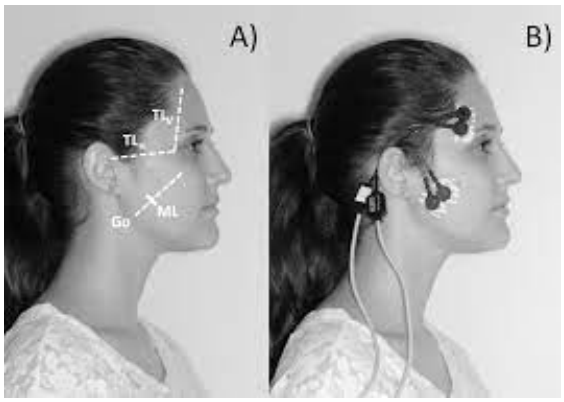
Because of the design of the earables to be placed in the ear, some of the participants had trouble fitting them inside of their ears. Because the audio playback or audio recording with the built-in microphone was out of the scope of this study, the earables were secured with skin-friendly tape.

6.3.2 EMG-Electrodes

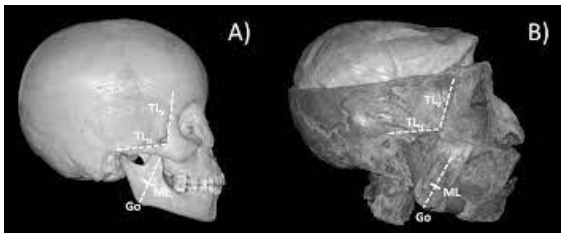
To collect EMG signals, special electrodes must be placed on the skin surface covering the target muscle. It is not a trivial task, as the anatomy of the participant dictates the optimal spot of the electrode placement. The following research paper was used as a reference to find the optimal EMG spots [14].

The following EMG electrode fig. 6.2 was used. For every target muscle, a pair of such electrodes is needed. Additionally, a reference electrode must be also placed on a neutral part of the head. The reference cables were fused together and connected to an electrode placed under the participant's chin.

The masticatory muscles are relatively small, so a slice of the adhesive foam was removed to bring the conductive parts of the two electrodes placed on one muscle, closer together. Even with this modification, the temporalis muscle was hard to capture, as even a misplace of 1mm from the optimal spot will introduce noise, or will capture the EMG signal of the nearby muscles (i.e. orbicularis muscle responsible for blinking).



(a) (A) Reference lines drawn on the face of a volunteer; (B) electrodes positioned over the AT and SM muscles. [14]



(b) (A) The lateral view of the skull shows the reference lines used in this study: TLV, TLH, and ML. The small thick line represents an intersection at 40% the length of ML from Go landmark; (B) Lateral view of deep facial planes, evidencing the AT and SM muscles and their relationship with anatomical landmarks and reference lines adopted. [14]

Figure 6.1: Optimal EMG spots for masticatory muscles by [14]

Figure 6.2: EMG electrode (57 x 34mm)

6.3.3 Throat Microphone

The throat microphone used in this study didn't have any adjustment possibilities. 2 of the participants had a smaller neck circumference, and as a result, it was not possible to capture the eventual vibrations that would propagate through the neck.

7. Code Availability

The complete code for the custom-built framework can be found under https://github.com/trupus/bruxism-benchmark

A. Plots

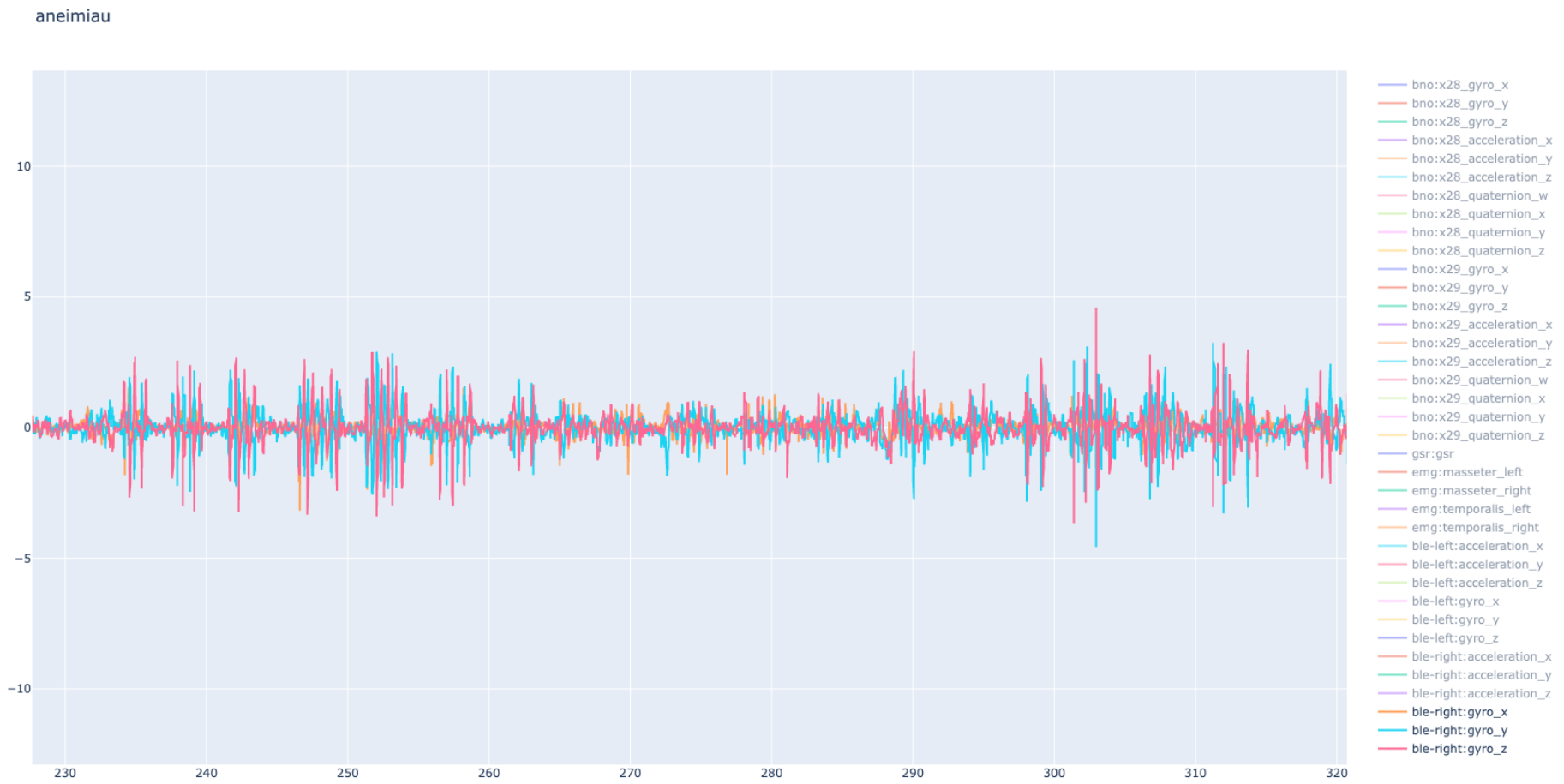


Figure A.1: Example of a right eSense earable gyroscope x, y, and z time-series slice. The x, y, and z plots are overlapped. From left to right: 3x2s grinding with the right side; 3x4s grinding with the right side; 3x2s grinding with the front side; 3x4s grinding with the front side; 3x2s grinding with all sides; 3x4s grinding with all sides

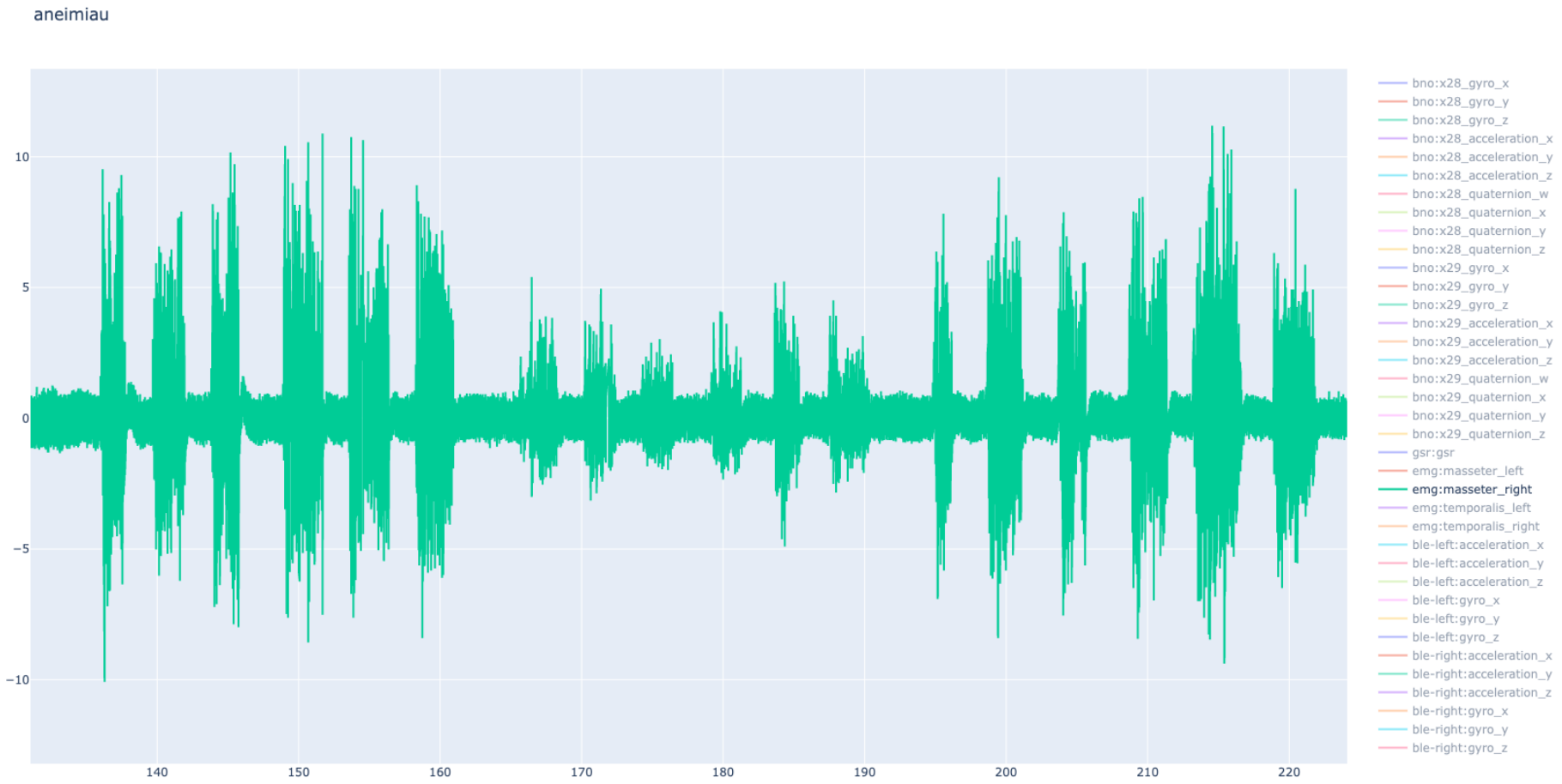


Figure A.2: Example of a right masseter time-series slice. From left to right: 3x2s clenching with the right side; 3x4s clenching with the right side; 3x2s clenching with the front side; 3x4s clenching with the front side; 3x2s clenching with all sides; 3x4s clenching with all sides

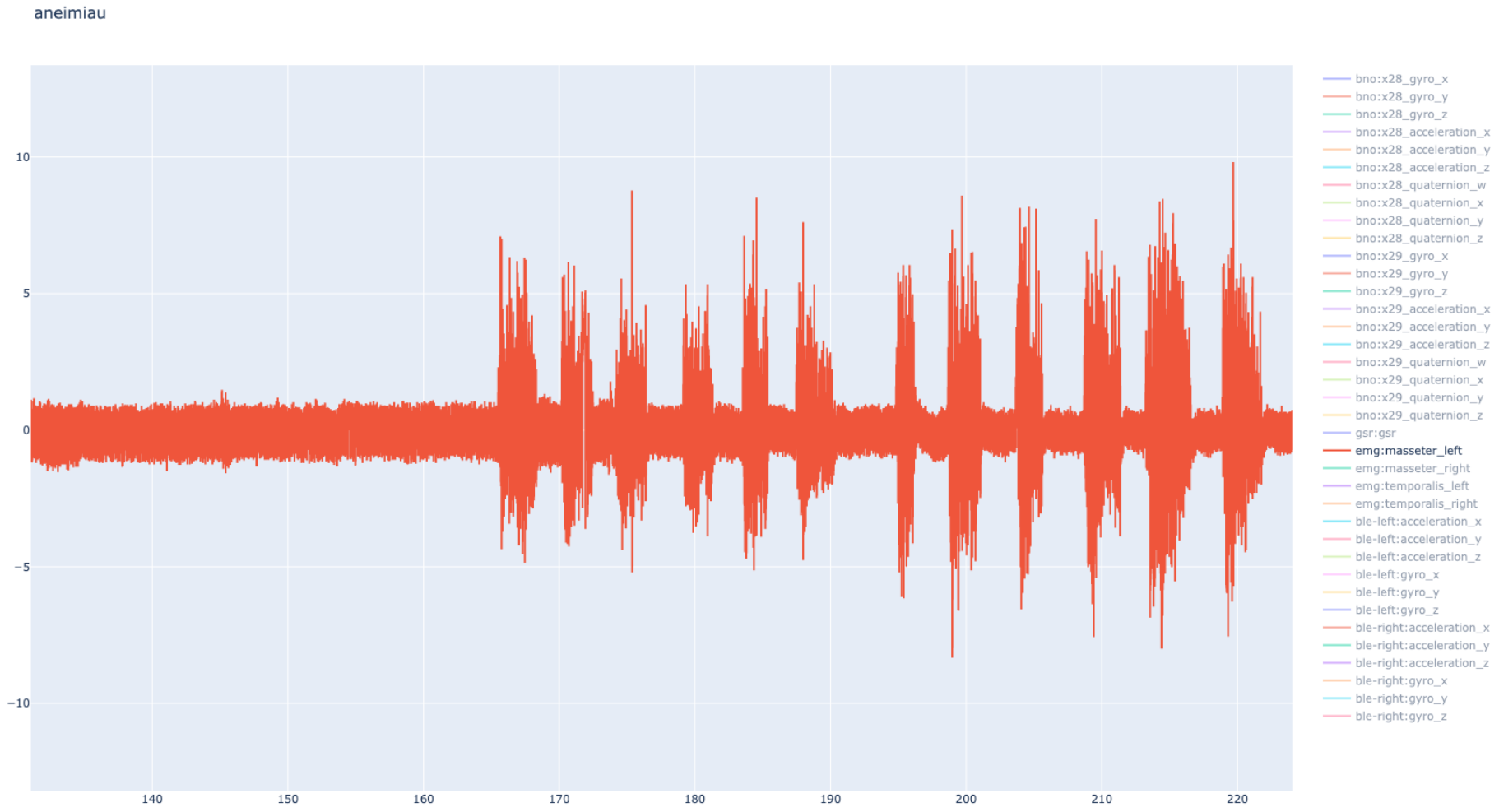


Figure A.3: Example of a left masseter time-series slice. From left to right: 3x2s clenching with the right side; 3x4s clenching with the right side; 3x2s clenching with the front side; 3x4s clenching with the front side; 3x2s clenching with all sides; 3x4s clenching with all sides

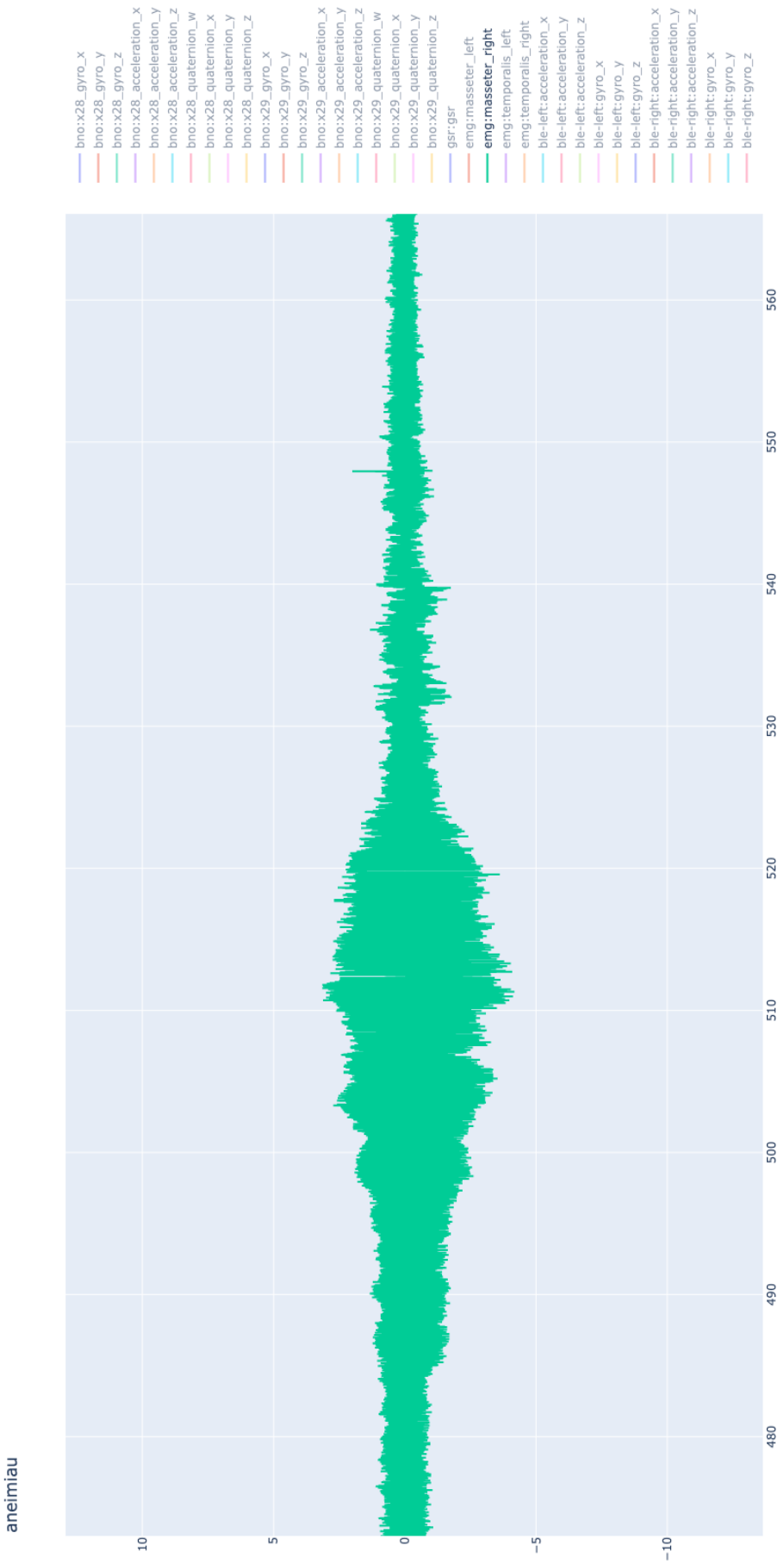


Figure A.4: Example of a right masseter time-series slice. Reading

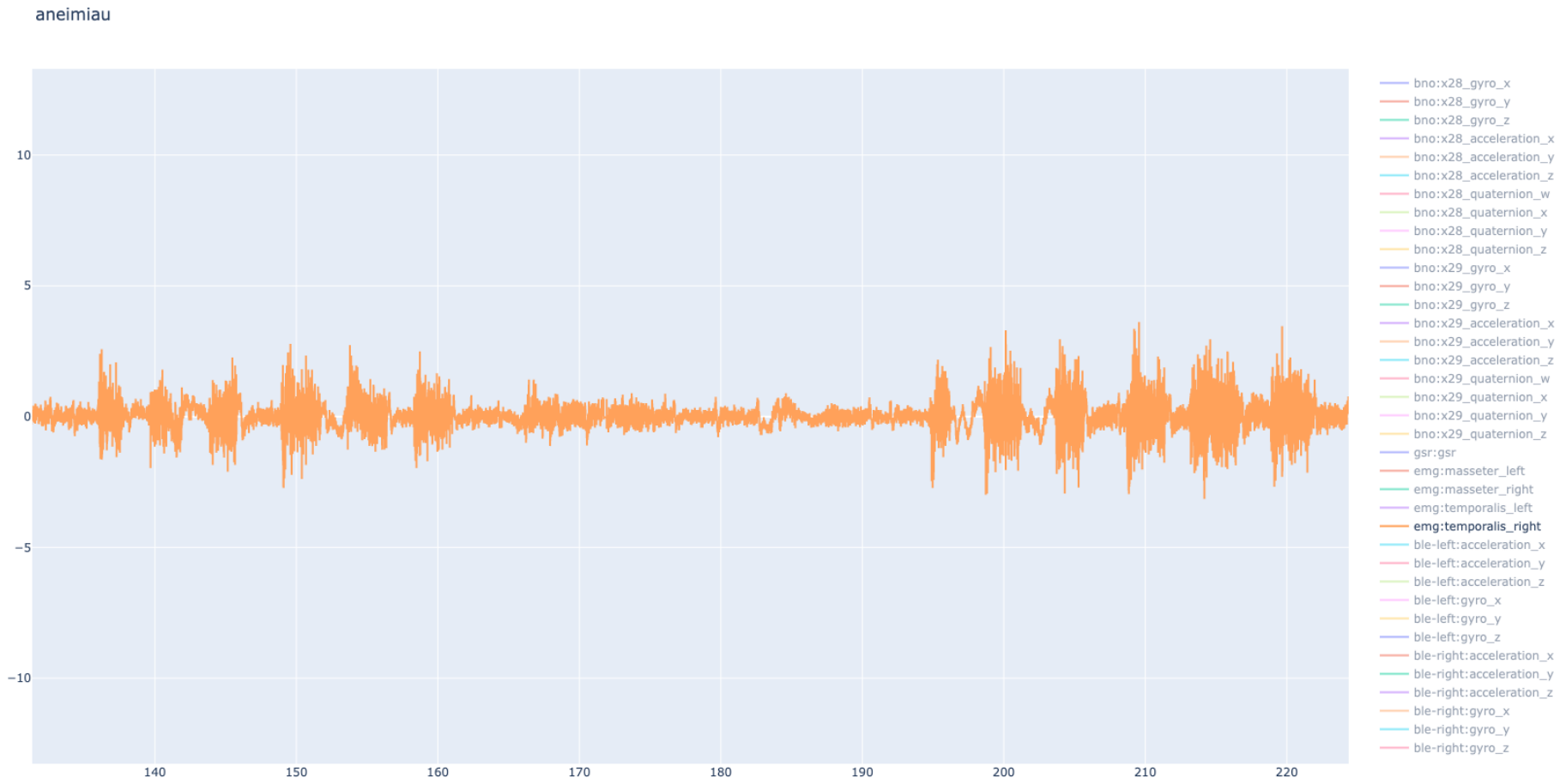


Figure A.5: Example of a right temporalis time-series slice. From left to right: 3x2s clenching with the right side; 3x4s clenching with the right side; 3x2s clenching with the front side; 3x4s clenching with the front side; 3x2s clenching with all sides; 3x4s clenching with all sides

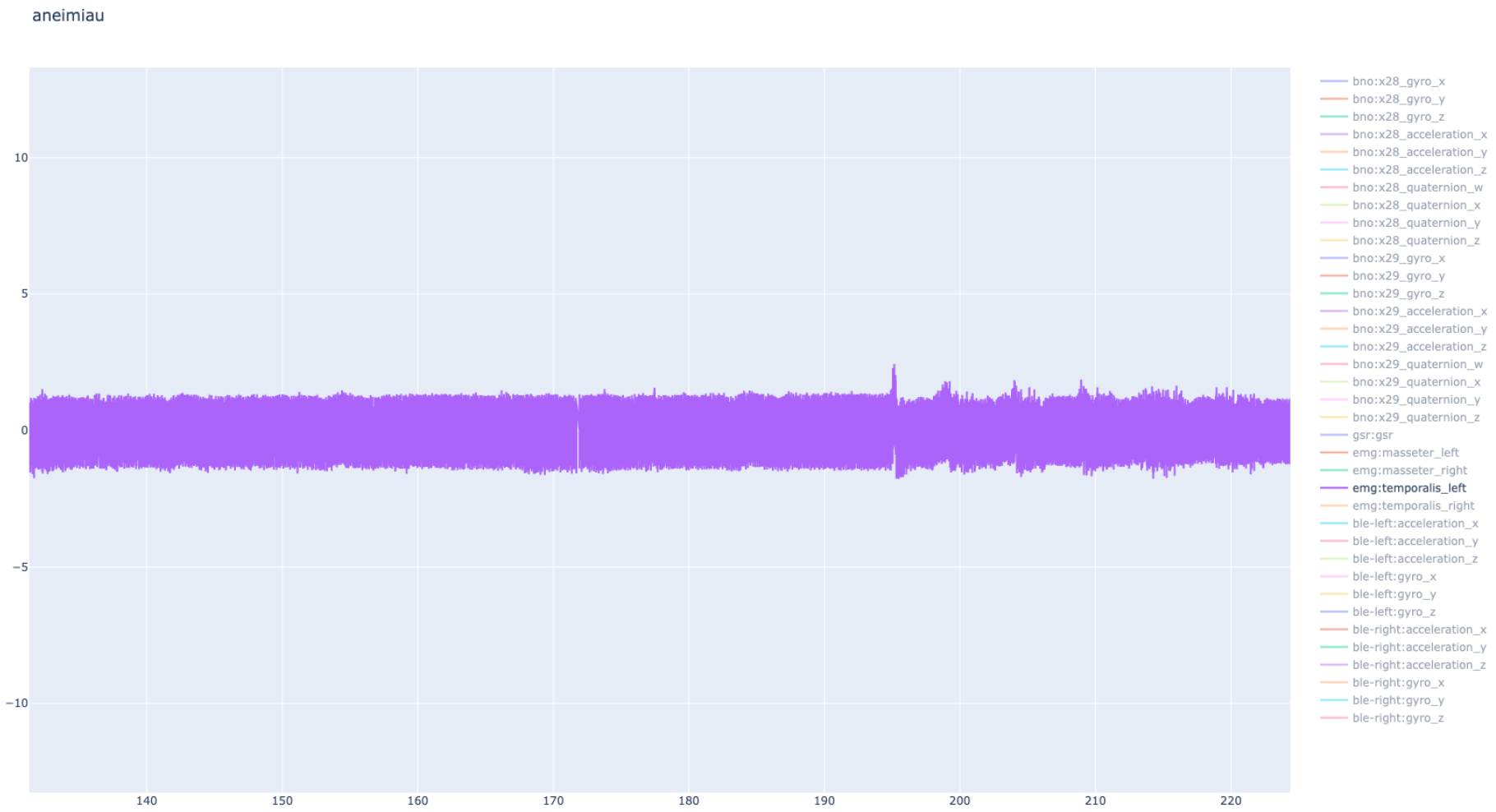


Figure A.6: Example of a left temporalis time-series slice. From left to right: 3x2s clenching with the right side; 3x4s clenching with the right side; 3x2s clenching with the front side; 3x4s clenching with the front side; 3x2s clenching with all sides; 3x4s clenching with all sides. Note that it's not possible to differentiate between the performed exercises here.

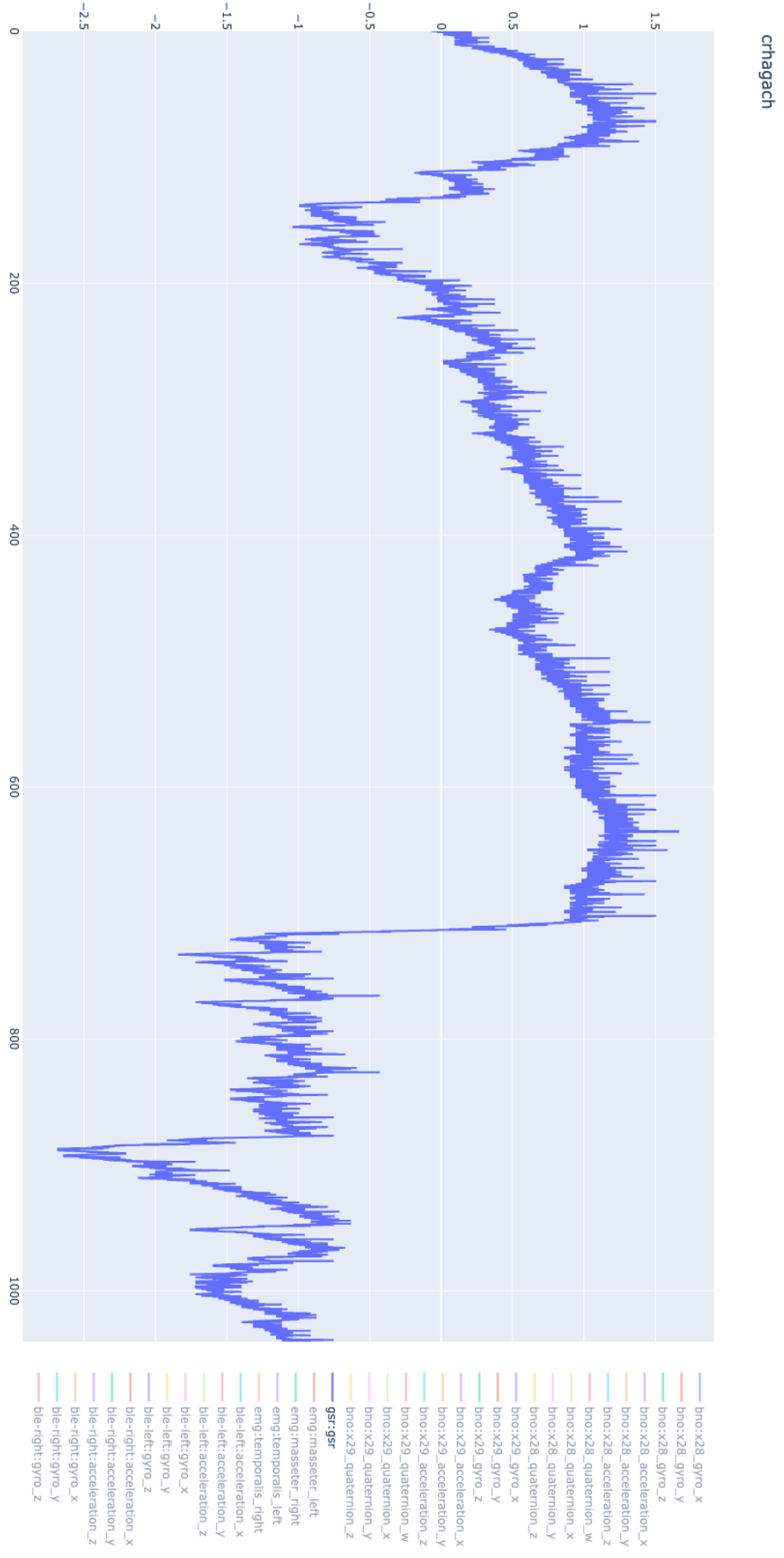


Figure A.7: Example of a GSR time-series

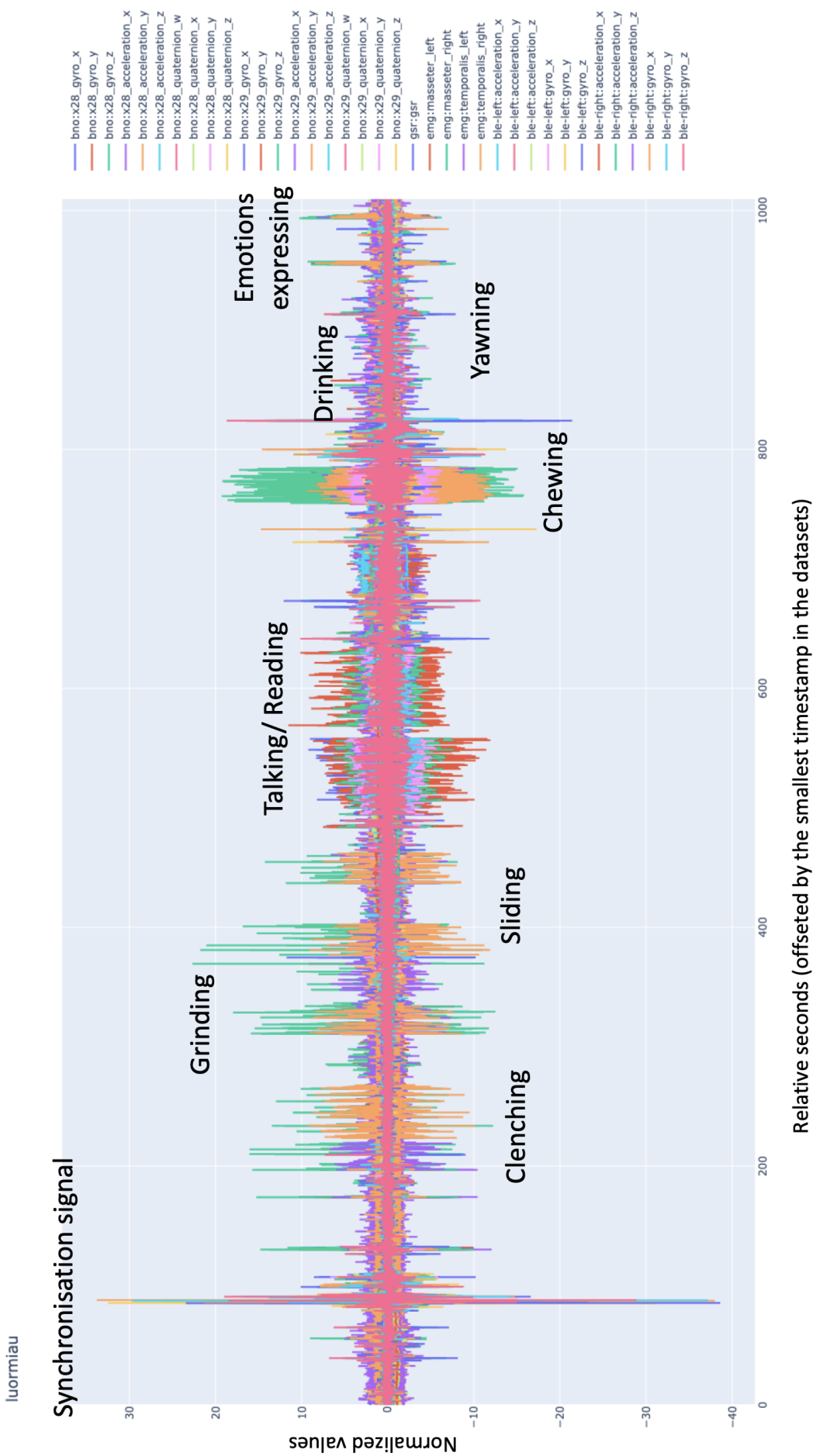


Figure A.8: Overview of a normalized dataset

Bibliography

[1] Camila Megale Almeida-Leite, Juliana Stuginski-Barbosa, and Paulo César Rodrigues Conti. "How psychosocial and economic impacts of COVID-19 pandemic can interfere on bruxism and temporomandibular disorders?" In: *Journal of Applied Oral Science* 28 (2020).

[2] Toshiyuki Ando et al. "CanalSense: Face-related movement recognition system based on sensing air pressure in ear canals". In: Association for Computing Machinery, Inc, Oct. 2017, pp. 679–689. ISBN: 9781450349819.

[3] Erika Bondareva, Elín Rós Hauksdóttir, and Cecilia Mascolo. "Earables for Detection of Bruxism: a Feasibility Study". In: *Adjunct Proceedings of the 2021 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2021 ACM International Symposium on Wearable Computers*. 2021, pp. 146–151.

[4] James Crook. *Audacity*. https://www.audacityteam.org/.

[5] Alona Emodi-Perlman et al. "Temporomandibular disorders and bruxism outbreak as a possible factor of orofacial pain worsening during the COVID-19 pandemic—concomitant research in two countries". In: *Journal of Clinical Medicine* 9.10 (2020), p. 3250.

[6] *Esense Ble Specifications*. https://www.esense.io/share/eSense-BLE-Specification.pdf.

[7] Fahim Kawsar et al. "Earables for personal-scale behavior analytics". In: *IEEE Pervasive Computing* 17.3 (2018), pp. 83–89.

[8] Michael Thomas Knierim, Max Schemmer, and Dominik Woehler. "Detecting Daytime Bruxism Through Convenient and Wearable Around-the-Ear Electrodes". In: vol. 275. Springer Science and Business Media Deutschland GmbH, 2021, pp. 26–33. ISBN: 9783030800901.

[9] Roya Lotfi et al. "A comparison between audio and IMU data to detect chewing events based on an earable device". In: *Proceedings of the 11th Augmented Human International Conference*. 2020, pp. 1–8.

[10] Daniel Lundqvist, Anders Flykt, and Arne Öhman. "Karolinska directed emotional faces". In: *Cognition and Emotion* (1998).

[11] Daniele Manfredini et al. "Epidemiology of bruxism in adults: a systematic review of the literature". In: *J Orofac Pain* 27.2 (2013), pp. 99–110.

[12] Chulhong Min, Akhil Mathur, and Fahim Kawsar. "Exploring audio and kinetic sensing on earable devices". In: *Proceedings of the 4th ACM Workshop on Wearable Systems and Applications*. 2018, pp. 5–10.

[13] Jay Prakash et al. “EarSense”. In: Association for Computing Machinery (ACM), Sept. 2020, pp. 1–13.

[14] Ana Sabaneeff et al. “Proposal of surface electromyography signal acquisition protocols for masseter and temporalis muscles”. In: *Research on Biomedical Engineering* 33 (2017), pp. 324–330.

[15] Shilpa Shetty et al. “Bruxism: a literature review”. In: *The Journal of Indian prosthodontic society* 10.3 (2010), pp. 141–148.

[16] Joanna Smardz et al. “Correlation between sleep bruxism, stress, and depression—a polysomnographic study”. In: *Journal of clinical medicine* 8.9 (2019), p. 1344.

[17] Temel Sonmezocak and Serkan Kurt. “Detection of EMG signals by neural networks using autoregression and wavelet entropy for bruxism diagnosis”. In: *Elektronika ir Elektrotechnika* 27 (2 2021), pp. 11–21. ISSN: 20295731.

[18] Wei Sun et al. “TeethTap: Recognizing Discrete Teeth Gestures Using Motion and Acoustic Sensing on an Earpiece”. In: Association for Computing Machinery, Apr. 2021, pp. 161–169. ISBN: 9781450380171.

[19] BA Thompson, BW Blount, and TS Krumholz. “Treatment approaches to bruxism.” In: *American family physician* 49.7 (1994), pp. 1617–1622.

[20] Adrian U Yap and Ai Ping Chua. “Sleep bruxism: Current knowledge and contemporary management”. In: *Journal of conservative dentistry: JCD* 19.5 (2016), p. 383.