Driver Drowsiness Detection using the Human-tenna Effect

BSc thesis

David Laclé
Mai Anh Nguyen
Harim Suleman

Technische Universiteit Delft

TU Delft

Driver Drowsiness Detection using the Humantenna Effect

BSc thesis

by

David Laclé
Mai Anh Nguyen
Harim Suleman

Project duration:	April 2025 – June 2025
Student number:	5246105 – David Laclé
	5224667 – Mai Anh Nguyen
	5014700 – Harim Suleman
Supervisor:	dr. H. Bastawrous
Thesis committee:	dr.ir. M.A.P. Pertijs
	dr.ir. M. Taouil
Date:	June 30, 2025

TU Delft

Preface

Drowsy driving contributes to a significant number of traffic accidents, yet it is often overlooked. Unlike alcohol impairment, the signs of drowsiness are subtle and frequently recognized too late. Its impact is severe, linked to an estimated 20 % of all traffic accidents and costing the Netherlands approximately 6 billion euros annually. The gradual onset of drowsiness makes timely detection challenging.
This project investigates a novel, non-intrusive method for detecting driver drowsiness by employing the Humantenna effect. By measuring grip strength through ambient electromagnetic signals, captured via sensors integrated into the steering wheel, and validating it with heart rate data, the system aims to passively detect drowsiness without sacrificing driver comfort or privacy.

Working on this project has been challenging and has provided valuable insights into safety-critical system design. We have gained a deep appreciation for the effort and complexity involved in developing systems that operate reliably in real-world conditions. We would like to sincerely thank our supervisor, Dr. Hany Bastawrous, for his guidance, continuous support, and constructive feedback throughout this work.
We would also like to express our gratitude to the hardware group, our colleagues, Matthieu Demeestere, Peter Nagy and Sanne Romijn, working towards the same goal, for their collaboration and valuable contributions throughout the project.

David Laclé
Mai Anh Nguyen
Harim Suleman
Delft, June 2025

Contents

1 Introduction

Every year, approximately 1.19 million people die from road accidents worldwide, while 20 to 50 million people suffer non-fatal injuries [1]. It is estimated that 20% of all road accidents in the Netherlands involves drowsiness [2], highlighting the need for effective detection and prevention strategies. However, it is hard to prove that drowsiness is the primary cause of an accident. In most cases, the accident is registered under a different cause, such as a red light negation or not yielding the right of way [2]. Therefore, this percentage is actually higher in reality.
A survey across 19 European countries aiming to estimate the prevalence and consequence of drowsy driving found that in 17% of the cases where drivers fell asleep, 7% resulted in an accident. 13.2% of the drivers required hospital care and 3.6% caused fatal accidents [3]. Not only does drowsy driving cause accidents, it also comes with economic consequences. While there is limited research about the specific costs related to accidents due to drowsiness, the total social cost of traffic accidents in 2022 in the Netherlands was estimated at 33 billion euros, equivalent to 3.4% of the national GDP [4]. This includes medical expenses, emergency services cost, legal fees, insurance handling, property damage, and lost productivity [4]. If drowsiness accounts for roughly 20 % of these crashes, that would correspond to over 6 billion euros in avoidable costs each year. Despite the notable economic consequences, the loss of human life is the most severe. Hence, drowsy detection systems are essential, in order to, prevent accidents involving drowsy driving.

Drowsiness is defined as an abnormal feeling of sleepiness or difficulty staying awake [5]. The most common causes of drowsiness are sleep deprivation, irregular sleep patterns, circadian rhythm disruptions or use of sedating medication [6]. Unlike fatigue, physical or mental exhaustion, drowsiness involves a strong urge to fall asleep. These conditions pose a significant risk to drivers by impairing reaction times, reducing situational awareness and compromising poor decision-making [7]. What distinguishes drowsiness from alcohol impairment is that the effects of alcohol are typically immediate and noticeable, whereas the signs of drowsiness develop gradually and go unnoticed. As a result, a drowsy driver may remain unaware of their impaired state until it is too late. This further emphasizes the need for early and reliable detection methods.

1.1. Current technologies

To address this problem, various driver sensing systems have been developed, which can be categorized into three types, as shown in Table 1.1: physiological, behavioral and vehicle-based.
However, only behavioral and vehicle-based systems are widely implemented in modern vehicles. Behavioral systems utilize cameras to detect signs of drowsiness by analyzing the driver's facial features. These systems depend on light conditions. Variations in lightning affect the reliability of the detection and in the worst case no detection becomes possible. Moreover, behavioral systems pose privacy concerns as many features are extracted from the face [8] as in shown in Table 1.1. Vehicle-based systems rely on onboard sensors such as light detection and ranging (LiDAR), steering-angle encoders, cameras on the exterior of the car and accelerometers to monitor changes in driving behavior that may indicate drowsiness. Although, many of these systems do not rely on lightning conditions and demonstrate a

higher reliability by combining multiple parameters, they only work in steady traffic conditions where the vehicle travels at speeds exceeding 65 km/h. As driving patterns are easier to monitor at these speeds and deviations can be attributed to drowsiness.

Table 1.1: Performance of existing drowsiness detection methods where 1 denotes the best performing method and 3 denotes the worst performing method [9]

Metric	Physiological	Behavioural	Vehicle-Based
Performance	1	2	3
Physical Intrusiveness	3*	1	1
Psychological Intrusiveness	1	3	2
Privacy Concern	1	3	2
Data Size	2	3	1
System cost	3	2	1
Computational cost	2	3	1
Susceptibility to noise	3	1	2
Vulnerability to Subject Diversity	2	3	1
Adaptability	1	3	2

* This can vary depending on the acquisition system.

While not yet implemented in current vehicles, a lot of research is going into measuring physiological changes in the body [10]. Behavioral changes follow from physiological ones and so measuring the changes directly would yield faster drowsiness detection. Examples of current physiological sensors are EEG [1], ECG [2], PPG [3] The main disadvantage of physiological systems is that the driver is required to wear the sensor directly on the body, this is reflected in Table 1.1 [9]. For the EEG this corresponds to helmet that makes direct electrical contact with the scalp. ECG requires electrodes on the skin and PPG requires a photosensor that measures light through the skin, usually the wrist. PPG has been successfully implemented in commercial products such as smartwatches and these products could potentially be developed into a minimally invasive physiological measurement system. Nevertheless, the driver is still required to wear the device at all times. In [10], the use of cameras, EEG, PPG and GSR[4] are compared by using single modality and multi-modality analysis. Figure 1.1 shows that the EEG is the most accurate for attention detection. While the accuracy of the other technologies are dependent on the subject. PPG and facial recognition alternate between being the second and third best and GSR performing the worst.

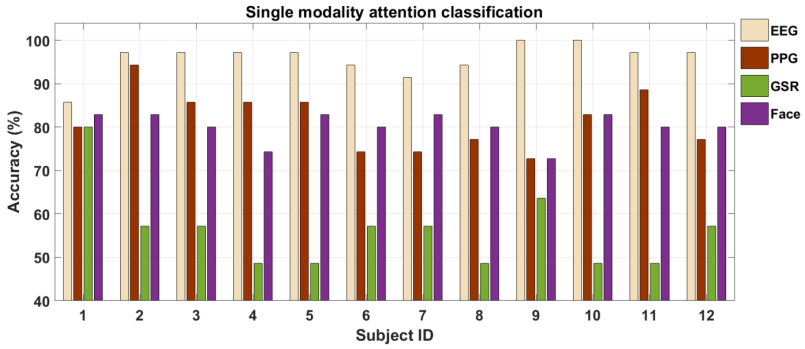


Figure 1.1: Single modality classification performance for driver attention analysis [10]

For current physiological measurement systems the driver must have direct contact with the sensors.

[1] Electroencephalography: measures electrical activity in the brain by using electrodes attached to the scalp [11].
[2] Electrocardiogram: records the electrical signals in the heart. [12]
[3] Photoplethysmogram: measures the photoresponse of blood. Provides the heart rate, blood oxygen saturation and other physiological parameters [13].
[4] Galvanic Skin Response: the change in skin conductivity due to sweat gland activity [14].

Requesting drivers to wear sensors before every trip is impractical. However, physiological measurements offer more accurate drowsiness detection with less computational demand compared to camera-based behavioral systems. This highlights the need for a non-invasive, cost-effective sensing method that ensures accuracy without requiring user compliance or expensive equipment.

1.2. Current commercially implemented technologies in cars

An overview of commercially implemented driver monitoring systems is provided in Table 1.2. These systems rely primarily on vehicle-based data to monitor driver behaviour. For activation, Volvo and Tesla require a minimum speed of 65 km/h for activation, while Mercedes sets the threshold at 60 km/h [15–17]. In Figure 1.2, Volvo's Driver Alert system is demonstrated. Additionally, the systems have listed limitations related to weather and road conditions such as strong side winds and uneven surfaces. The cameras and radars must remain unobstructed as well. Mercedes uses a driver camera to detect microsleep by analyzing the driver's eyes, although it is limited by poor ambient lightning and the type of spectacles and sunglasses being worn. When drowsiness is detected, the warning in Figure 1.3, is shown on the driver's display [17]. Tesla's cabin camera, shown in Figure 1.4, detects driver inattentiveness [16]. Both systems, however, raise privacy concerns due to continuous monitoring of the drivers behavior.

Table 1.2: Overview of Commercial Driver Monitoring Systems

Company	Product	System Type	Product Notes
Volvo [15]	Driver Alert	Vehicle-based	Radar, camera and steering wheel movements
Mercedes [17]	Attention Assist with microsleep detection	Vehicle-based, behavioural	Driver camera for microsleep detection
Tesla [16]	Driver Drowsiness Warning	Vehicle-based, behavioural	Tesla Vision*, cabin camera and cabin radar

*Tesla's camera-based Autopilot system replacing ultrasonic sensors [18].

(a) Driver Alert reads the position of the car in the lane.

(b) The car is being driven erratically in the lane.

Figure 1.2: Volvo's driver alert [15]

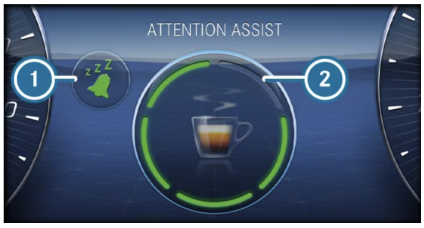


Figure 1.3: Mercedes' Attention Assist warning sign. 1 denotes the microsleep detection status. 2 denotes the detected attention level, the higher the detected attention level the more segments are colored green [17]

(a) The cabin camera is located above the rear view mirror in the Tesla Model 3

(b) The camera's field of view

Figure 1.4: Tesla's cabin camera [16]

1.3. Project objective

The primary objective of this research is to detect driver drowsiness by employing the Humantenna effect. This utilizes the human body as an antenna to detect ambient electromagnetic signals, specifically the 50 Hz electromagnetic fields emitted by power lines and electrical devices. The effect enables the detection of the presence and movement of the driver, as well as providing the ability to estimate grip strength. The principles of the Humantenna effect are further elaborated in section 3.2. Grip strength will be used as an indicator to assess whether a driver is drowsy [19]. In [20], drowsiness could be detected from monitoring the heart rate using a ECG. The proposed method to determine drowsiness from heart rate in [20] will be used to confirm drowsiness from grip strength by monitoring both grip strength and heart rate in an experiment. The two proposed algorithms will be used in this research.
The system will operate in the background, a steering wheel sleeve equipped with two sensors at where the hands will be placed will be used, as shown in Figure 1.5. This means the driver does not need to wear any sensors or devices. The system will therefore be non-obstructive to the drive experience. By using passive sensing through the Humantenna effect, the system will require minimal modifications to current vehicles. As the system operates by detecting only ambient electromagnetic signals, no personal data will be collected from the driver, ensuring privacy.
Ultimately, this research aims to improve road safety by providing a non-obtrusive drowsiness detection system that can easily be adopted in vehicles.

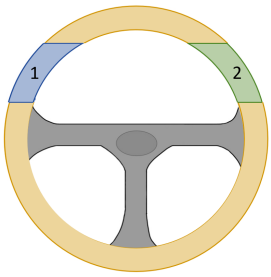


Figure 1.5: Steering wheel with yellow sleeve with hand placement

1.4. Thesis structure

Chapter 2 specifies the program of requirements. In the next chapter 3 the focus is on the theory and background for this thesis. Drowsiness, humantenna effect and a system overview will be given. The methods proposed by [20] use multivariate statistical process control (MSPC). This will be explained in this chapter as well. In chapter 4 the methods and implementation for validating grip strength will be discussed. In addition to this, the data collection of the grip strength and heart rate will be discussed. The next chapter 5 follows this by explaining how the data collected from the method in the previous chapter is used to determine drowsiness.

2

Program of requirements

As a full product is not yet being developed, there are no strict requirements in this project. This is a exploratory project, investigating whether it is feasible to use the human-antenna effect to detect drowsiness. Therefore, the project remains in the research phase, and considerations such as power efficiency and other constraints relevant to a finished product are not yet being addressed.

2.1. Mandatory requirements

2.1.1. Non-functional requirements

1. The system should maintain consistent drowsiness detection accuracy across users with varying grip strengths and genders

2.1.2. Functional requirements

1. The system should relate the RMS voltage of the grip sensor signal to estimated grip force using a calibrated model.
2. The system should simultaneously sample and record at least three analog sensor channels, with two dedicated to grip sensing and one to the heart signal, at a minimum sampling rate of 1000 Hz per channel.
3. The system is able classify the user's drowsiness state every 60 seconds based on extracted features.
4. The system should trigger a visual or audio alert if drowsiness is detected in 2 or more consecutive windows.
5. The system should store all recorded sensor data in CSV format for offline analysis.

2.2. Trade-off requirements

1. Sensor's accuracy versus its cost.

3

Theory and Background

3.1. Drowsiness

Drowsiness, as stated in the introduction, is commonly defined as an abnormal feeling of sleepiness or difficulty staying awake [5]. Thus, drowsiness is the stage before actual sleep takes place. Sleep consists of two main phases: REM (rapid eye movement sleep), characterized by vivid dreaming and active brain waves, and NREM (non-REM sleep), which involves deeper, restorative sleep. NREM sleep can be categorized into three stages, namely the N1, N2 and N3 stage [21]. The N1-stage is characterized as the transition phase between wakefulness to sleep. Importantly, N1 is also associated with microsleeps, which are brief, involuntary episodes of sleep that last just a few seconds. Microsleeps occur when a persons brain momentarily shifts into sleep-like activity despite the person appearing awake. These brief moments of inattentiveness are significant in contexts such as driving, as they increase the risk of accidents. In the N1 stage, muscle activity remains relatively intact, eye movements become slow and intermittent, and individuals can be easily awakened by external sensory stimuli [20]. Therefore, for the system to successfully warn a drowsy driver with a sound alert, it is important to detect drowsiness during or preferably just before the N1 stage.

Drowsiness is most accurately measured using electroencephalography (EEG), which can directly detect changes in brain activity that correspond to different stages of alertness and sleep. However, EEG systems require multiple electrodes to be placed on the scalp and demand a complex setup and calibration, which can interfere with the driving experience and are impractical for both experimental and real-world deployment in vehicles.
In addition to physiological measurements, the Karolinska Sleepiness Scale (KSS) [22] is commonly used to measure the level of drowsiness at a particular time of the day. The KSS is a 9-point self-report scale shown in Table 3.1. The scale ranges from 1 (extremely alert) to 9 (very sleepy, great effort to keep awake, fighting sleep) The KSS was validated with the EEG and the electro-oculographic (EOG)[1] [22]. The KSS has been used in sleep research and real-time drowsiness assessments to capture participants own perception of sleepiness [24]. It provides a straightforward and reliable method for linking self-reported alertness levels with physiological measures in experimental contexts, such as driving simulations.
The autonomic nervous system (ANS) and cardiac activities are affected when there are changes in sleep condition [25]. The autonomic nervous system (ANS) controls a wide range of involuntary physiological functions, such as heart rate, breathing, digestion, and pupil response. It is composed of two primary branches: the sympathetic nervous system (SNS) and the parasympathetic nervous system (PNS). During wakefulness and periods of heightened alertness, sympathetic activity tends to increase, contributing to elevated heart rate and reduced heart rate variability(HRV) due to more consistent cardiac rhythms. Heart rate variability (HRV) refers to the fluctuations in the R-R intervals (RRIs), which are the time intervals between successive R-waves in an electrocardiogram (ECG). As drowsiness

[1] The electro-oculogram (EOG) is a record of the electrical dipole occurring between the front and the back of the eye and reversing in current direction when the eye moves from side to side. The height of the potential difference increases in conditions of bright illumination. [23].

sets in, parasympathetic activity becomes more prominent, promoting a slower heart rate and a relative increase in HRV. This physiological shift reflects a reduction in arousal and cognitive engagement, making HRV a sensitive and non-invasive indicator of declining alertness. The HRV reflects the activity of the ANS [26], allowing for drowsiness detection by measuring the heart rate through an ECG. In this study, electrocardiography (ECG) was chosen as a more practical alternative for indicating drowsiness during the experiment. Although ECG does not measure brain activity directly, changes in heart rate variability (HRV) are known to correlate with different levels of alertness [27]. Compared to EEG, ECG requires fewer sensors, is easier to set up, and is less intrusive making it more suitable for extended data collection sessions in a simulated driving task.

Table 3.1: Karolinska Sleepiness Scale (KSS)

Score	Description
1	Extremely alert
2	Very alert
3	Alert
4	Rather alert
5	Neither alert nor sleepy
6	Some signs of sleepiness
7	Sleepy, but no effort to keep awake
8	Sleepy, some effort to keep awake
9	Very sleepy, great effort to keep awake, fighting sleep

3.2. Physical Principle Behind the Proposed Sensor

Electromagnetic (EM) waves, from 50 Hz power-line fields to AM/FM radio, mobile networks and Wi-Fi, are present throughout our environment but remain invisible to the naked eye. Any conductive object in these fields will pick up a small voltage, and the human body is no exception. The body can be modeled as a short dipole about 1.7 m tall at wave length of $l \leq \frac{\lambda}{50}$ this is only valid for frequencies lower than $f_{max} = 3.5$ MHz [28]. The body develops an open-circuit voltage proportional to the field strength and its effective length. This human-antenna effect has been used to detect presence and movement [29,30], and more recently to estimate grip strength [19]. In this study a simplified model is considered of the human body as a wire of length of L, this results in to an induced voltage $V_{ind}(t)$ in equation 3.1.

$$V_{\text{ind}}(t) = \int_0^L \mathbf{E}_{\text{inc}}(\mathbf{r}) \cdot d\ell \;\approx\; E_{\text{inc}}\, L_{\text{eff}} \cos\theta \;\sin(\omega t) \tag{3.1}$$

where E_{inc} is the magnitude of the incident electric field, L_{eff} the effective conductor length projected onto the field, and θ the angle between the field and the conductor axis. In the quasi-static region (wavelength $\lambda \gg L$), this reduces to a simple linear relationship $V_{\text{ind}} \approx E_{\text{inc}} L_{\text{eff}}$. No resonant behavior or radiation patterns occur, since L is electrically short.

The sensor chosen for the steering wheel is an insulated wire. This choice was made by the hardware subgroup. Their work showed that an insulated wire is much more sensitive to changes in grip, because it forms a capacitive coupling with the hand. In contrast, a bare wire simply shorts the signal and shows almost no change, since its impedance is negligible compared to the input impedance of the amplifier. This leads to the following model, the power line and human body form a fixed capacitance $C_{line\text{-}human}$, and the human body and sensor form a variable capacitance $C_{human\text{-}sensor}$. These two capacitors are connected in series. The human stays in a fixed position in these experiments, and thus $C_{line\text{-}human}$ is treated as a constant and only $C_{human\text{-}sensor}$ will vary with grip strength. The sensors capacitance is considered in this case as two parallel plate capacitor and is approximated by equation 3.2.

$$C_{\text{human-sensor}} \;=\; \varepsilon \frac{A}{d} \tag{3.2}$$

where ε is the permittivity between the hand and wire, A is the effective overlap area (which varies slightly with hand position), and d is the distance between hand and wire (which decreases under applied pressure). In this study the total area of the wire is designed such that it fits completely in the palm, so that A can be treated as a constant and $C_{human-sensor}$ primarily becomes a function of d. The whole system can be modeled by two capacitance in series, this can be modeled by equation 3.3.

$$C_{eq} = \frac{C_{line-human} \cdot C_{human-sensor}}{C_{line-human} + C_{human-sensor}} \tag{3.3}$$

From the hardware sub group findings the $C_{line-human}$ is 100 pF and coupling is $C_{human-sensor}$ is 0.66 pF and thus $C_{human-sensor}$ will dominate the total capacitance.The complete circuit representing this model can be seen in Figure 3.1. The voltage that is measured at the sensor output can be represented by equation 3.4.

$$V_{sensor} = \frac{Z_{input}}{Z_{input} + Z_{eq}} \cdot V_{source} \tag{3.4}$$

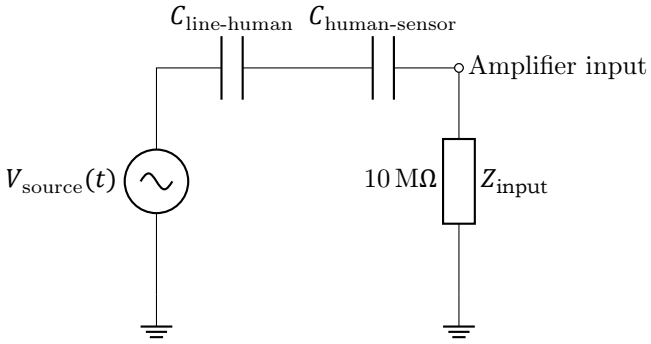


Figure 3.1: Complete circuit model from the power line to the input of the amplifier

3.3. System Overview

The full system can be divided into eight components, taking into account both physical sensing and digital processing. This thesis primarily focuses on the latter stages: digitization, recording, feature extraction. See Figure 3.2.

- Environmental EM Field: 50 Hz ambient electromagnetic field originating from power lines.
- Human Body Coupling: The human body acts as a receiving antenna and couples with the EM field.
- Capacitive Sensor: A wire on the steering wheel forms a capacitive coupling with the hand.
- Amplifier: The weak AC signal is amplified and centered around 2.5 V for ADC compatibility.
- ADC (MCP3208): Converts the analog signal to a 12-bit digital signal at 1000 Hz per channel.
- ESP32 Microcontroller: Manages ADC communication and streams data over USB.
- PC Recording Interface: Stores data in CSV format for offline processing.
- Feature Extraction & Classification: RMS features of the induced voltage and HRV metrics are computed for use in drowsiness detection.

The ESP32 was selected for this project due to its low cost relative to its performance and its ease of use, particularly in rapid prototyping environments. Initially, the internal ADC of the ESP32 was used. However, it exhibited significant non-linearity due to being an early hardware revision and was unable to sample multiple channels simultaneously at high frequencies without introducing crosstalk and signal drift. To address these limitations, the external ADC (MCP3208) was chosen, which offered improved performance, higher accuracy, and more reliable multi-channel sampling. Due to time constraints, the amplifier was not ready in time for testing, so a two-channel Digilent oscilloscope for the final signal acquisition was used. The Digilent API was employed to automate acquisition and stream both sensor channels directly into Python for recording. Heart rate data continued to be captured by the ESP32, since the oscilloscope only supports two input channels. All of the Python scripts and ESP32 C++ code used in arduino IDE for data collection are provided in Appendix B.

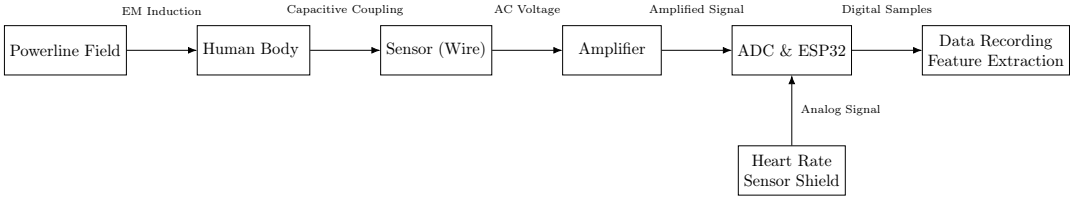


Figure 3.2: Complete signal acquisition pathway, including human antenna effect and heart rate sensor input

4

Methods and Implementation

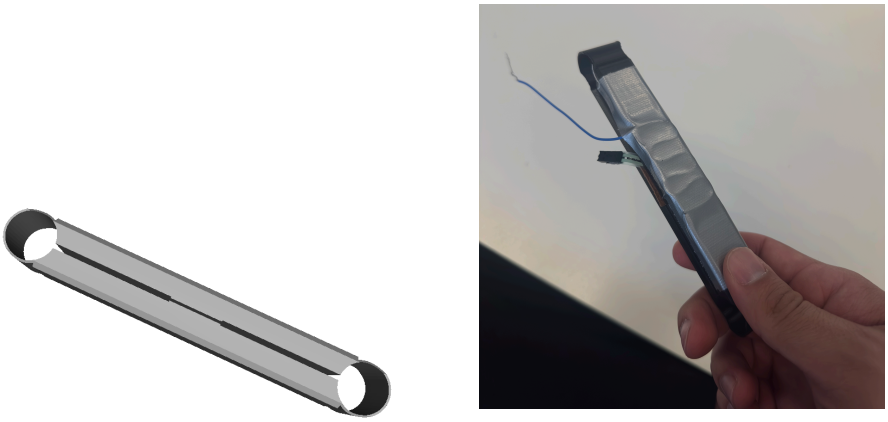
This chapter establishes the methodology for validating a novel human-antenna sensor capable of measuring grip strength. First, the experimental setup and calibration procedure used to compare the antenna-based sensor against a reference resistive force sensor are presented. Next, the signal-processing pipeline is examined in detail, from induced body voltage to feature extraction. Finally, the driving-simulation experimental setup is described.

4.1. Grip strength validation

Grip strength is inversely correlated with drowsiness [9], and can be monitored unobtrusively via the human-antenna effect. Prior work in [19] demonstrated quantification of discrete grip levels on a steering wheel using this phenomenon. In this study, the continuous grip signal is captures from an isolated wire antenna embedded in the wheel, while simultaneously recording a calibrated resistive force sensor as ground truth. By synchronizing and correlating the antenna-based voltage RMS with the reference pressure measurements, the reliability of the noninvasive antenna method as a proxy for actual grip force is evaluated.

4.1.1. Experimental Setup

Both sensors are mounted on a custom 3D-printed handle Figure 4.1. The insulated wire is embedded on the outer surface to measure induced voltage, while the reference force sensor sits in between the handle to record applied grip force. During each trial, participants start with a relaxed grip and then steadily increase their force until reaching their maximum.



(a) 3D model prototype render

(b) Grip strength prototype showing an insulated wire sensor on top

Figure 4.1: Grip strength prototype showing an insulated wire sensor on top and the force sensor in between

4.1.2. Reference Pressure Sensor Characterization

To measure applied force, a strain gauge made from thin metallic foil was used. When force is applied, the foil deforms, changing its effective area and thus its resistance. The sensor was selected because of its compact size, its 0-5 kg range and cost-effectiveness. To derive force from the sensor output, first the weightresistance calibration curve was establisheed by recording resistance values under known weights see Figure 4.2. The log-log plot of these data reveals an approximately linear relationship, so a power-law model was fitted, shown in equation 4.1.

$$R_{\text{sensor}}(w) = a\,w^{b} \tag{4.1}$$

The calibration curve was fitted using a power-law model by applying a linear regression on the log-transformed data, using Pythons `np.polyfit(log_weights, log_resistances, 1)` function. The fit quality was evaluated based on the root-mean-square error (RMSE), which corresponds to an accuracy of approximately $\pm$ 30 g. However, during repeated testing we observed up to $\pm$ 100 g variation, likely due to hysteresis and drift. Nevertheless, the sensor reliably captures small relative changes in grip force, which is critical for the goal of tracking trends in grip strength rather than measuring absolute force precisely.

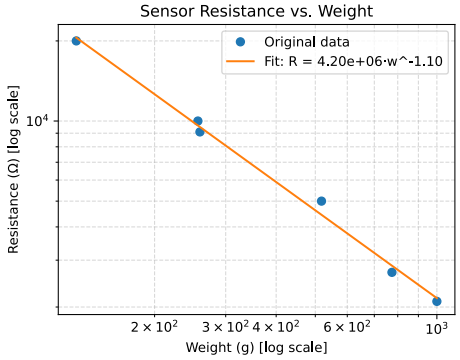
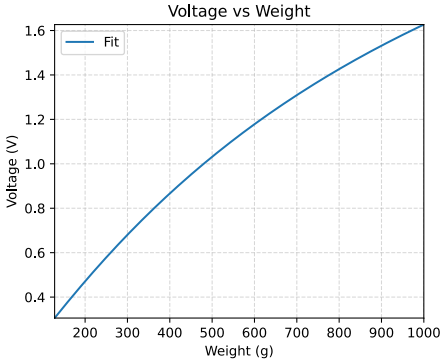
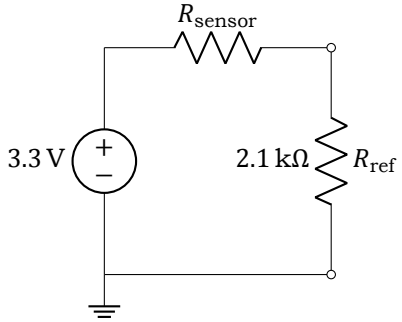


Figure 4.2: Divider resistance vs weight

A voltage divider configuration is used to convert the variable resistance of the force sensor into a measurable voltage. In this setup, a known reference resistor R_{ref} is connected in series with the sensor, and the voltage is measured across R_{ref}. Given the supply voltage and the known resistance value, the sensors resistance can be calculated from the measured voltage. Using the previously fitted model described by equation 4.1, the resistance value can then be converted into an estimated applied force in grams. This makes it possible to map voltage readings over R_{ref} directly to grip force measurements as shown in Figure 4.3a. The complete circuit used for this conversion is shown in Figure 4.3b.



(a) Voltage vs Weight model created based on power-law model



(b) Voltage divider circuit used to read the force resistance sensor

Figure 4.3: Voltage vs Weight model created based on the voltage output of voltage divider and power-law model

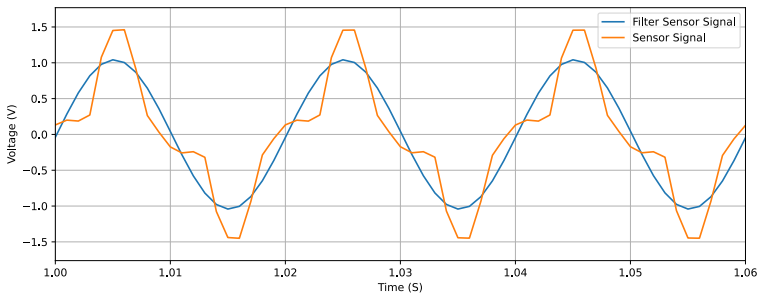
4.1.3. Grip Strength Induced Voltage

At the output of the designed sensor, a 50 Hz signal and its higher order harmonics is observed. See Figure 4.4. This signal originates from the electromagnetic field generated by power lines. Previous research by Microsoft [30] has demonstrated that it is possible to classify human movement by analyzing the voltage induced on the body from these ambient fields. In that work, electrodes were directly attached to the body to measure the induced voltage, and features such as the DC level, RMS voltage, and frequency components were extracted for classification. In our case, the sensor is capacitively coupled to the hand. This setup eliminates the possibility of measuring DC component as this coupling between hand and sensor behaves like a high-pass filter and block the DC component. As a result, the analysis is limited to the RMS value of the 50 Hz component, which provides a robust measure of signal energy.

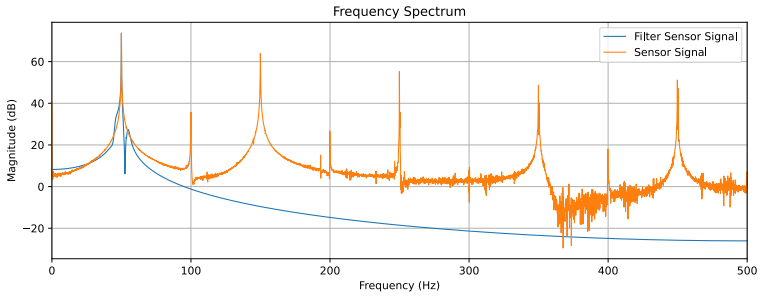
Filtering

As stated in the previous section, only the 50 Hz component will be used for further analysis. Therefore, filtering is required to isolate this component. While a 500 Hz anti-aliasing filter is present in the analog front-end, implemented by the hardware group, additional filtering must be performed in the digital domain.

In digital signal processing, two common types of filters are Finite Impulse Response (FIR) and Infinite Impulse Response (IIR) filters. For this application, the key requirements include minimal ripple, fast settling time, and computational efficiency. As discussed in [31], IIR filters offer a faster response, significantly lower ripple, and reduced computational cost compared to FIR filters of equivalent order. The primary drawback of IIR filters is their potential instability if not carefully designed, as well as their non-linear phase response. However, since the signal of interest contains only a single frequency component, phase distortion is not a concern, making IIR filters a suitable choice for this application. Therefore, a Butterworth IIR bandpass filter is selected for its maximally flat passband. Using SciPys signal.butter the filter can be design to meet the following specifications: a 45-55 Hz passband, a 40-60 Hz stopband, and at least 20 dB of stopband attenuation. These requirements yield a 5th order Butterworth design that cleanly isolates the 50 Hz component with minimal distortion.



(a) Induced Voltage (Time Domain)



(b) Induced Voltage (Frequency Domain)

Figure 4.4: Raw signal and filtered signal measured from the body using the capacitive sensor.

Feature extraction

To quantify the energy of the 50 Hz signal, a root mean square (RMS) value is computed over sliding window. The RMS is defined as:

$$\mathrm{RMS}[k] = \sqrt{\frac{1}{N}\sum_{i=0}^{N-1} x[k+i]^2} \tag{4.2}$$

where $x[n]$ is the sampled signal and k indexes the start of each window. After computing $\mathrm{RMS}[k]$, the window is shifted by Δn samples. by setting $\Delta n = 1$, this result in total amount of sample $k = L - N + 1$, since the $L \gg N$ this gives roughly 1 RMS value per sample which make it easier to plot and align with the original signal. This gives overlap of 99.8 % samples between successive computations.
Choosing an appropriate window size is a trade-off between noise suppression and responsiveness to signal changes:

- Small windows (e.g., 100 ms) offer high temporal resolution but may produce noisy features due to insufficient averaging.
- Large windows (e.g., 1 s) provide smoother estimates but may obscure short-term changes and transitions in grip or alertness.

After trail and error, a window size of 500 ms was selected. This length strikes a balance between stability and responsiveness and is sufficient to capture multiple cycles of the 50 Hz signal within each window.

4.2. Data Collection

To predict driver drowsiness based on grip strength signals recorded from the steering wheel, it is important to note that while a decrease in grip strength can sometimes indicate drowsiness, it may also simply reflect a relaxed but alert state. Therefore, to accurately distinguish between these scenarios and identify true signs of drowsiness, some type of classification model has to be explored. One way can be to use a supervised machine learning model trained on labeled grip strength data. Another way is to use a statistical model to do classification. In both cases, using such a model requires a dataset that includes grip strength signals along with corresponding labels indicating the drivers state, whether alert or drowsy. As no publicly available dataset meets these specific criteria, a custom dataset was created under controlled conditions as part of this study.
This section describes the experimental setup used to collect the data, the participants involved, and the methodology used to assign ground-truth labels based on drowsiness, which were derived using ECG-based HRV features and Multivariate Statistical Process Control (MSPC).

4.2.1. Experimental Setup

To collect realistic grip strength data related to drowsiness, a simulated driving environment was created. The goal was to mimic actual driving conditions while maintaining enough control to ensure clean and consistent signal acquisition. This approach allows for safe induction of drowsiness-related behavior without putting participants at risk.
As this study relies on the Humantenna effect to measure grip strength, the signal is susceptible to small environmental changes. Variations in surrounding electromagnetic fields, body positioning, and even nearby objects can influence the captured signal. Therefore, to ensure consistency and minimize noise, data collection was conducted under controlled conditions. A schematic of the experimental setup is given in Figure 4.5.

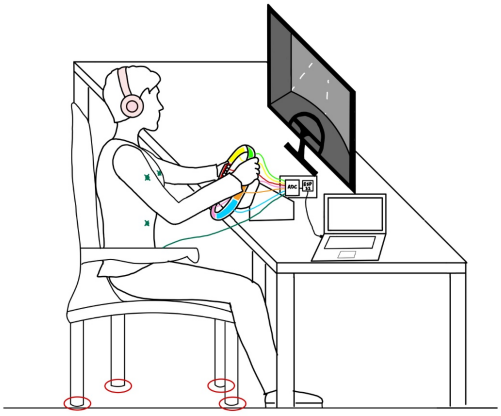


Figure 4.5: Experimental setup

The experiment was conducted in an indoor environment, where a gaming steering wheel was securely mounted to a desk. Tape markers were placed on the floor to mark the position of the chair and prevent variations in body positioning. Additionally, wires near the steering wheel were taped down to keep them stable and avoid introducing noise due to movement. Two insulated wires were attached to the left and right sides of the steering wheel, as shown in Figure 4.6. These positions were chosen based on the standard hand placement recommended during driving, specifically, the 10 and 2 hand position. This was to ensure natural and consistent contact with the sensors while driving. Each sensor on the steering wheel is connected to a separate channel of the oscilloscope which is then connected to the laptop through usb to read and save the data. The actual setup is presented in Figure 4.7.
In parallel, an ECG sensor was connected to an ESP32 microcontroller using an ECG shield. Electrodes were placed on the upper left, upper right, and lower left part of the chest. ECG signals were recorded

simultaneously with the grip signals and synchronized using timestamps to allow accurate alignment of the physiological data with variations in grip strength. The ECG recordings were later used to derive HRV features for objective drowsiness labeling.

(a) Front on the steering wheel (b) Back of the steering wheel

Figure 4.6: Illustration of how the wires are attached to the steering wheel to measure grip strength signals

Figure 4.7: Experimental setup

The experiment was divided into two phases: a baseline phase and a drowsiness induction phase. In both phases, volunteers participated in a driving simulation game which is designed to replicate typical driving scenarios while maintaining participant engagement.
At the start of the baseline phase, a grip calibration procedure was performed. This step was essential, as individual participants can have significantly different grip strengths. During this procedure, participants were asked to apply light, medium, and maximum grip to the steering wheel. These calibrated values were recorded and later used as features in the data set to help the machine learning model interpret the absolute grip strength values in the context of the individual range of each participant. Subsequently, the ECG data were recorded during a 15-minute period during which the participant was instructed to remain alert, to obtain a clear baseline to compare against.
During the drowsiness induction phase of the experiment, participants were instructed to play relaxing music through their headphones while continuing the driving simulation. A highway scenario was selected for the driving course, as highway driving is typically monotonous and more likely to induce drowsiness. This phase aimed to simulate realistic conditions under which drowsiness might naturally occur. This phase had a duration of 60-90 minutes. An overview of the experimental phases is given in Table 4.1

Table 4.1: Overview of the two experimental phases

Phase	Activities	Purpose
Baseline Phase	Grip calibration (light, medium, max) Driving simulation with alert state 15-minute ECG recording	Establish participant-specific grip strength range and collect baseline ECG/HRV features during alertness.
Drowsiness Induction Phase	Relaxing music via headphones Monotonous highway driving simulation 60-90 minute duration	Create conditions conducive to natural drowsiness for comparison against baseline data.

4.2.2. Participants and Ethical Considerations

To ensure model generalized well across the population, a diverse group of participants was included in the data collection. Variations in factors such as gender and baseline grip strength were considered, as they can significantly influence grip strength measurements and how drowsiness manifests. A dataset reflecting this diversity is important as it reduces the risk of model bias toward any specific subgroup and improves robustness for real-world applications.
Due to time constraints and the proof-of-concept nature of the study, the sample size was limited. Nonetheless, this initial dataset laid the groundwork for future studies to expand participant diversity to further enhance model performance.

The inclusion criteria required participants to have no known cardiovascular or neurological diseases, as such conditions could affect ECG signal interpretation. Participants also needed to be physically capable of operating the driving simulator safely. After the experiments procedures were explained, each participant signed an informed consent form provided in Appendix A.1 acknowledging data anonymity and voluntary participation. To further control for external factors influencing drowsiness, participants completed a short questionnaire assessing regular sleep patterns, general medical health, and other potential interfering factors such as recent caffeine or alcohol intake. These responses helped contextualize individual variations in grip behavior during data analysis while ensuring adherence to ethical screening standards. The questionnaire is also provided in Appendix A.2. None of the participants were rejected in this study, as everyone slept more than 6 hours and were not flagged corresponding to no health liabilities that may influence the HRV. For this study, 6 participants were recruited. Table 4.2 shows the participants' key characteristics, including age, gender, and sleep patterns, helping to define the target population for these findings.

Table 4.2: Participant characteristics

Characteristic	Value
Total Participants	N = 6
Gender Distribution	3 Male, 3 Female
Average Sleep Duration	6.5 $\pm$ 1.2 hours (mean $\pm$ SD)

5

Labeling Methodology

It is essential to obtain ground-truth labels that accurately reflect the driver's drowsiness state during the experiment. Several approaches were considered for labeling. One option was to periodically ask participants to self-report their level of drowsiness during the driving simulation using the Karolinska Sleepiness Scale (KSS). However, this method was ultimately rejected, as it introduces potential for human error and could interfere with the natural progression of drowsiness by repeatedly drawing the driver's attention away from the task. Instead, a more objective method was selected, namely to determine the state of the driver using ECG-derived physiological signals.

5.1. ECG-Based Labeling Approach

To generate reliable drowsiness labels from ECG signals, the method, proposed by Fujiwara et al. in their study, Heart Rate Variability-Based Driver Drowsiness Detection and Its Validation With EEG [20] was adopted. Their work demonstrated a strong correlation between HRV features and drowsiness states, which were validated using EEG. In this study, their approach is replicated by first extracting HRV features from the ECG signal and then applying a multivariate statistical process control (MSPC) method to detect deviations associated with drowsiness. An important aspect of this approach is the use of a baseline segment, recorded while the participant is fully alert. This baseline serves as a reference for normal physiological behavior. MSPC then treats drowsiness as an anomaly. An anomaly is a statistically significant deviation from this baseline. The following subsections describe how the ECG data from participants during the baseline and subsequent baseline and subsequent driving periods were preprocessed and analyzed.

5.1.1. Filtering

ECG data from participants were recorded continuously during both baseline and driving sessions. The raw signals contain noise and artifacts, which must be removed before further analysis. In Figure 5.1 the raw ECG data of the first 10 seconds of an arbitrary subject can be seen.
The first step in the filtering process was removing the baseline wander, which is low-frequency noise in the range of 0 to 0.5 Hz. The baseline wander is caused by respiration and body movements. To achieve this, a bandpass Butterworth filter with cutoff frequencies of 0.5 Hz and 20 Hz was applied. This configuration preserves the components of the ECG signal relevant for R-peak detection while suppressing noise from both low- and high-frequency sources. After the filtering the raw and filtered ECG data is plotted in Figure 5.1.

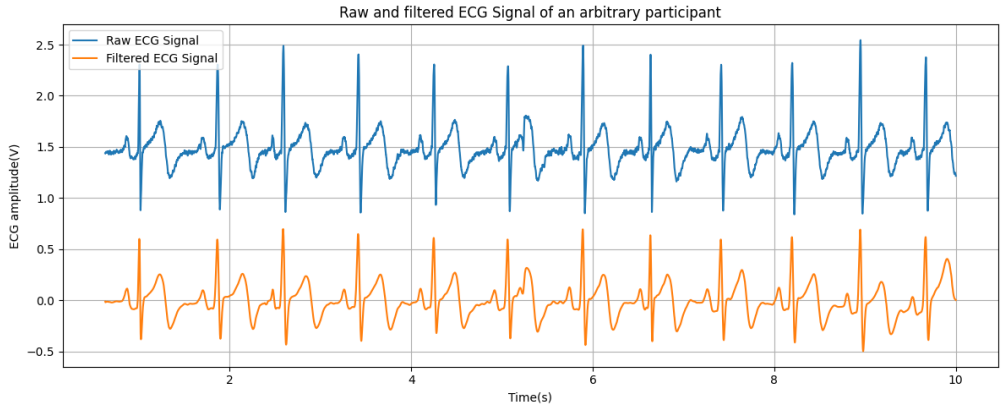


Figure 5.1: Comparison of raw and filtered ECG signal. During the filtering the baseline wandering, noise and DC offset were removed and the result is illustrated by the orange line

5.1.2. R-peak detection

After the noise and artifacts of the signal are filtered out, the R-peaks must be detected for further analysis. The human heart consists of four chambers: the upper chambers are called the atria, and the lower chambers are called the ventricles. Each heartbeat generates a distinct waveform in the ECG signal, typically comprising the P wave, QRS complex, and T wave as illustrated in Figure 5.2. The P wave represents atrial depolarization, which occurs when the atria receive an electrical signal and contract. The QRS complex corresponds to ventricular depolarization, indicating electrical activation and contraction of the ventricles. Finally, the T wave reflects ventricular repolarization, during which the ventricles relax and reset in preparation for the next cardiac cycle.

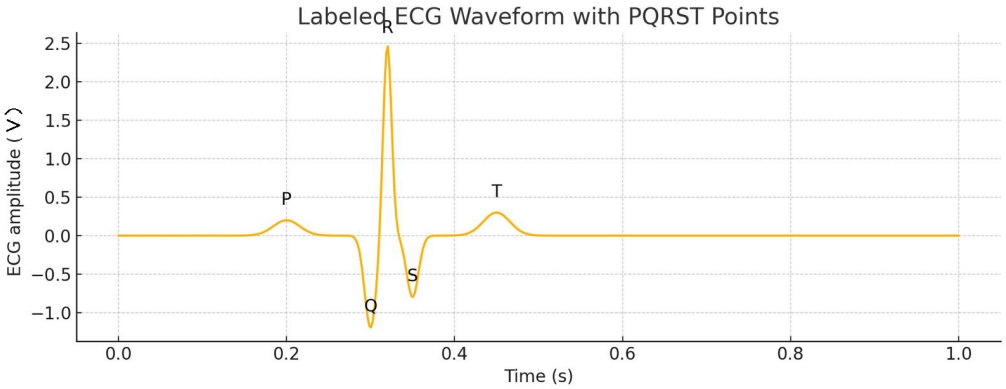


Figure 5.2: ECG waveform showing a single heartbeat, with labeled P wave, QRS complex, and T wave. Each segment corresponds to a specific phase of electrical activity in the heart

To extract useful features from the ECG signal, the QRS complex is of great importance, as it is the most easily detectable part of the waveform. Within this complex, the R peaks stand out as the highest-amplitude points. The R peaks correspond to the moment when the hearts ventricles contract and pump blood. This event produces the largest amplitude in the ECG waveform, making R peaks the easiest and most reliable points to detect. As heartbeats are marked by R peaks, the time intervals between them (RRIs), directly represent the hearts rhythm. This enables the study of heart rate variability (HRV) and, consequently, the assessment of autonomic nervous system activity and physiological states such as drowsiness.

For the detection of the R-peaks, the Pan-tompkins method was used, which is one of the most widely used methods for Rpeak detection. This algorithm demonstrated over 99% accuracy in detecting QRS complexes on the MITBIH Arrhythmia database [32]. This method differentiates and squares the signal to identify the high slope of the R-peaks. Finally, a moving window integrator is applied. This step

averages the values of the signal within a short window, effectively smoothing the waveform and further enhancing the detection of the R-peaks. The result of applying the Pan-tompkins method is given in Figure 5.3.

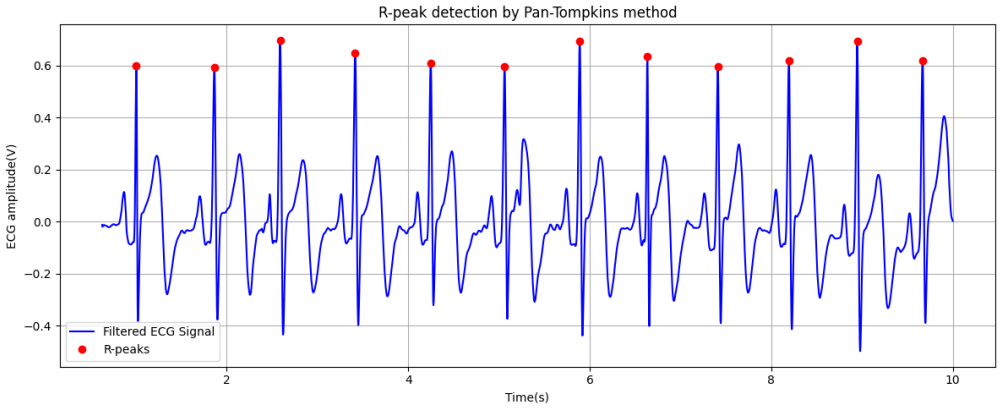


Figure 5.3: Pan-Tompkins method applied on the filtered ECG signal

5.1.3. HRV-feature extraction

HRV represents the fluctuation in the time between consecutive R-peaks (RR intervals) in the ECG signal, and these fluctuations reflect changes in the bodys physiological state, particularly in relation to alertness and drowsiness. HRV features are typically divided into linear and nonlinear categories. Following the methodology described in [20], only linear features were extracted, as nonlinear methods often require longer RRI recordings for reliable estimation, making them less suitable for short-term or real-time drowsiness detection. Among the linear features, both time-domain and frequency-domain measures were extracted to capture different aspects of heart rate variability. Features were extracted from 3-minute time windows, as this time frame has been reported to provide better drowsiness detection [33]. The characteristics of heart rate variability (HRV) are listed in Table 5.1.

Table 5.1: Extracted HRV features and their descriptions

Feature	Description	Domain
Mean NN	Mean of RR-intervals	Time
SDNN	Standard deviation of RRI	Time
RMSSD	Root mean square of the difference of adjacent RRI	Time
Total power (TP)	Variance of RRI	Time
NN50	Number of successive RR interval pairs that differ by more than 50 ms.	Time
LF Power	Power in low-frequency band (0.04–0.15 Hz)	Frequency
HF Power	Power in high-frequency band (0.15–0.4 Hz)	Frequency
LF/HF Ratio	Ratio of low to high-frequency power	Frequency

The Mean NN is the average time between consecutive R peaks in the ECG signal, calculated after removing abnormal beats and artifacts. During drowsiness, the Mean NN tends to increase, showing that the heart rate is slowing down. This happens because the parasympathetic nervous system becomes more active, which slows the heart and signals the body to relax. Tracking this change can help detect when someone is getting drowsy. The formula of calculating the Mean NN is given in Eq.5.1, where RR_i is the duration between the i-th and $(i+1)$-th R peaks, and N is the total number of RR intervals in the time window.

$$\text{Mean NN} = \frac{1}{N}\sum_{i=1}^{N} RR_i \tag{5.1}$$

SDNN is the standard deviation of all RR intervals in a given time window. Its calculated by first measuring all the intervals between consecutive R peaks, and then computing the standard deviation of those values in the time window as given in Equation 5.2, where $\overline{\mathrm{RR}}$ is the mean of all RR intervals. The SDNN gives a sense of how much the intervals vary over time. A higher SDNN means the heart rate is fluctuating more, which generally reflects a healthier and more alert state. During drowsiness, SDNN tends to decrease, as the bodys physiological state becomes more stable and less reactive.

$$\mathrm{SDNN} = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}\left(\mathrm{RR}_i - \overline{\mathrm{RR}}\right)^2} \tag{5.2}$$

RMSSD (Root Mean Square of Successive Differences) measures the short-term variability between consecutive RR intervals. It is calculated by taking the square root of the mean of the squared differences between adjacent RR intervals in a time window as shown in Equation 5.3. Unlike SDNN, which looks at all the variation in the RR intervals over a longer period, RMSSD focuses specifically on the quick, beat-to-beat changes. This makes RMSSD a good measure of short-term fluctuations in the heart-rate.

$$\mathrm{RMSSD} = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N-1}\left(\mathrm{RR}_{i+1} - \mathrm{RR}_i\right)^2} \tag{5.3}$$

The Total Power (TP) is the overall variance of the RR intervals within a time window. It indicates how much the heart rate fluctuates in total, combining both fast and short-term changes. In other words, it shows how much the RR intervals spread out around their average. A higher TP indicates that the heart is actively adjusting its rhythm, which is common when someone is alert. When drowsiness sets in, this variability usually decreases, and TP drops, showing that the heart rhythm becomes more steady and less responsive. The formula of calculating the Total Power is given by Equation 5.4

$$\mathrm{TP} = \mathrm{Var}(RR) \tag{5.4}$$

The NN50 counts how many times the difference between two consecutive RR intervals exceeds 50 milliseconds. In a relaxed or drowsy state, the time between heartbeats tends to change gradually, with differences often staying below 50 milliseconds. A difference greater than 50 milliseconds is considered a more significant shift in heart rhythm. This threshold was chosen empirically and later adopted as a standard in HRV analysis [34]. When the NN50 is high, the heart rate is fluctuating more strongly which can occur in both alert and drowsy states.

LF Power (Low-Frequency Power) reflects the variability in heart rate within the 0.04 to 0.15 Hz frequency band. It is calculated as the area under the power spectral density (PSD) curve of the RR intervals within the frequency band as given in Equation 5.5. The LF band reflects both sympathetic and parasympathetic activity, but it is often linked to how the heart rate adjusts when blood pressure changes. When someone is awake or under mild stress, the LF power is usually higher, showing that the body is actively responding to its environment. However, as a person starts to feel drowsy, the LF power tends to drop or vary [35].

$$\text{LF power} = \int_{0.04}^{0.15} P(f), df \tag{5.5}$$

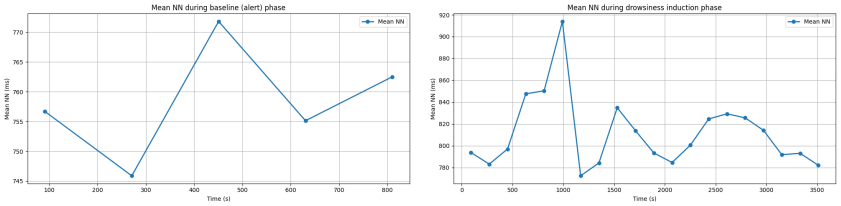
The HF Power (High-Frequency Power), measured in the 0.15 0.4 Hz range, indicates how the heart rate changes based on the breathing patterns. When someone is relaxed and breathing naturally, this value is usually higher. During drowsiness, the nervous system becomes less responsive and HF power can drop. The formula of the HF Power is given in Equation 5.6

$$\text{HF Power} = \int_{0.15}^{0.4} P(f)\, df \tag{5.6}$$

The LF/HF ratio reflects the balance between the sympathetic and parasympathetic branches of the autonomic nervous system as given in Equation 5.7. A higher ratio indicates alertness, whereas a lower ratio indicates drowsiness.

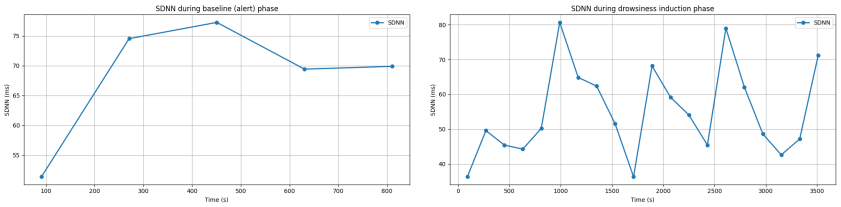
$$\text{LF/HF ratio} = \frac{\text{LF power}}{\text{HF power}} \tag{5.7}$$

Figure 5.4 and 5.5 presents the plots of the time-domain and frequency-domain features for an arbitrary participant during the baseline and the drowsiness induction phase. During the experiment, it was observed that in the first 15 minutes, the participant became drowsy and closed their eyes. This is also reflected in the drowsiness plots, particularly in the mean NN given, which increases around that time, indicating a drop in heart rate. The other features, such as SDNN, RMSSD, and NN50, also show noticeable changes or spikes at this point. Throughout the rest of the drowsiness induction phase, the features continue to fluctuate, suggesting variability in the participants physiological state as they alternated between drowsy and more alert periods. This is also observed in the frequency domain features such as the LF and HF plots. In the LF/HF ratio plot there are some spikes indicating alertness, but also some parts where the ratio drops well below the value of three, again indicating drowsiness. After the experiment, it was also confirmed by the participant that during some periods she felt drowsier than other periods.



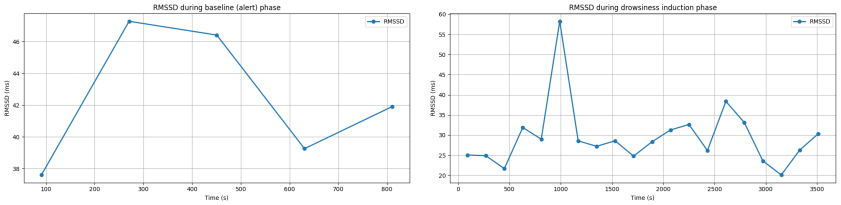
(a) Mean NN - Baseline

(b) Mean NN - Drowsy



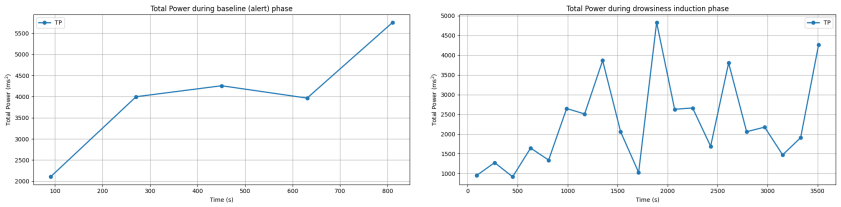
(c) SDNN - Baseline

(d) SDNN - Drowsy



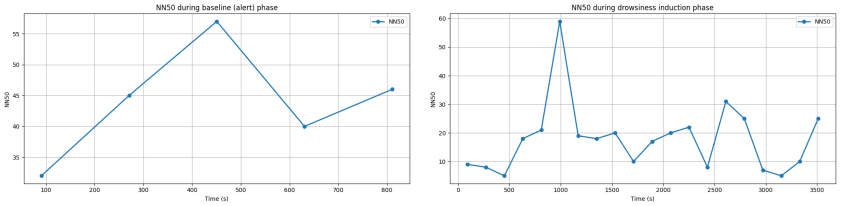
(e) RMSSD - Baseline

(f) RMSSD - Drowsy



(g) Total Power - Baseline

(h) Total Power - Drowsy



(i) NN50 - Baseline

(j) NN50 - Drowsy

Figure 5.4: Comparison of time-domain HRV features during baseline and drowsy states

(a) LF Power- Baseline (b) LF Power - Drowsy

(c) HF Power - Baseline (d) HF Power - Drowsy

(e) LF/HF - Baseline (f) LF/HF - Drowsy

Figure 5.5: Comparison of frequency-domain HRV features during baseline and drowsy states

5.1.4. MSCP

Univariate Statistical Process Control (USPC) is a simpler and commonly used method; however, it does not account for the correlation between variables. In other words, it cannot detect anomalies that arise from unusual combinations of otherwise normal individual values as those in heart rate variability (HRV) data. In Figure 5.6a, the star is not flagged as an anomaly because it lies within the rectangular region defined by USPC, which monitors each variable independently. If the ellipsoid area is used instead, the anomaly would have been detected.

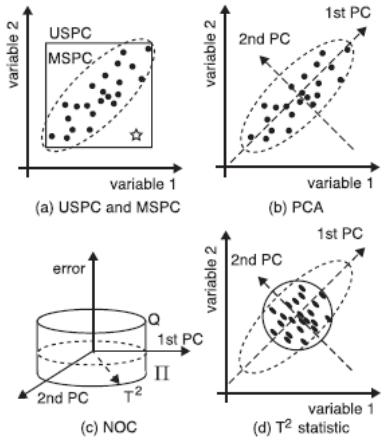


Figure 5.6: Schematic diagram of MSPC [20]

Therefore, Multivariate Statistical Process Control (MSPC) is used to monitor abnormalities caused by sleepiness across the eight HRV features listed in Table 5.1. This is done based on the approach of [20]. MSPC models the correlation structure among these features using Principal Component Analysis (PCA) [36]. PCA identifies new variables, called principal components, which are linear combinations

of the original features and capture the major trends (directions of maximum variance) in the dataset as shown in Figure 5.6b
First the HRV data are put into matrix $X \in \mathbb{R}^{N \times M}$, where each of the N rows represents an observation (such as a time window of HRV data), and each of the M columns corresponds to one of the HRV features. The data matrix is mean-centered so that each feature has a zero mean. Next Singular Value Decomposition (SVD) is used to extract the major trends from the data, this is applied to the matrix X. This decomposes X into three matrices seen in equation 5.8. U is the left singular matrix. Σ is the diagonal matrix with singular values as diagonal elements. V^T is the right singular matrix.

$$X = U\Sigma V^T = \begin{bmatrix} U_R & U_0 \end{bmatrix} \begin{bmatrix} \Sigma_R & 0 \\ 0 & \Sigma_0 \end{bmatrix} \begin{bmatrix} V_R & V_0 \end{bmatrix}^T \tag{5.8}$$

$$X = T_R V_R^T + E \tag{5.9}$$

Factorizing equation 5.8 by PCA gives equation 5.9 where $T_R \in \mathbb{R}^{N \times R} = U_R \Sigma$ this is the score matrix and contains the projections of the original observations onto the retained principal components. This approximation captures the most significant structure of the data, while filtering out noise and minor variations. The residual part of the data is the error matrix, $E \in \mathbb{R}^{N \times M}$. This matrix captures deviations from the learned multivariate pattern. Column V_R spans the subspace II in Figure 5.6, which shows the correlation among the variables.
To detect drowsiness-related anomalies, MSPC computes the Hotellings T^2 statistic, which measures how far a sample lies within the PCA subspace, as shown in equation 5.10. In this formula t_r is the score of the r-th principal component, σ_r is the corresponding standard deviation, and x is the new sample. This is equivalent to a Mahalanobis distance which is the distance normalized by the standard deviations of the scores, resulting in a region that is circular and nominal like in Figure 5.6d. The Q statistic, defined in equation 5.11, quantifies the deviation of the sample outside the subspace. Here, x_m is the original feature value, and $\hat{x}_m$ is the value reconstructed from the PCA model. The projection matrix $(I - V_R V_R^T)$ captures how far the observation deviates from the learned structure. Together, the T^2 and Q statistics define the Normal Operating Condition (NOC) of the system. Since these two metrics are based on orthogonal subspaces, the NOC region can be visualized as a cylinder: the radius corresponds to the control limit of T^2, and the height to the control limit of Q. If either statistic exceeds its control limit, the observation is flagged as anomalous, indicating a potential drowsy state.
The control limits for T^2 and Q are usually set based on confidence levels, such as 95% or 99%, depending on how sensitive the detection needs to be. To apply MSPC effectively, the number of principal components R must be chosen carefully. A common method is to select R so that the first R components explain around 80% to 90% of the total variance in the data.

$$T^2 = \sum_{r=1}^{R} \frac{t_r^2}{\sigma_{tr}^2} = x^T V_R \Sigma_R^{-2} V_R^T x \tag{5.10}$$

$$Q = \sum_{m=1}^{M} e_m^2 = \sum_{m=1}^{M} (x_m - \hat{x}_m)^2 = x^T (I - V_R V_R^T) x \tag{5.11}$$

5.1.5. Algorithm 1: Drowsy Detection Preparation

After extracting the features for each participant using three minute time segments, the drowsy detection preparation algorithm proposed by [20] could be implemented. This algorithm uses the principle MSCP 5.1.4. Before drowsiness could be detected, it was important to first determine how the HRV features behave during the alert phase. This was performed by implementing algorithm 1. Then, in algorithm 2, the MSCP was applied again but on the drowsiness induction data. Whenever a sample differed from the trend of the baseline recordings, the algorithm flagged this as drowsiness. The steps of the first algorithm that uses the baseline data as reference are given in Table 5.2

Table 5.2: Drowsy Detection Preparation Algorithm [20]

	Algorithm 1: Drowsy Detection Preparation
1	For all i such that $1 \leq i \leq I$ do:
2	Extract the ith driver's awake HRV feature $\tilde{X}^{(i)}$ from the ith driver's awake RRI data $y^{(i)}$.
3	End for
4	Merge matrices $\tilde{X}^{(1)}, \ldots, \tilde{X}^{(I)}$ into one matrix $\tilde{X}$.
5	Preprocess $\tilde{X}$ to obtain matrix X.
6	Derive Σ_R and V_R from X using Equation (1).
7	For all i such that $1 \leq i \leq I$ do:
8	Define the control limits of the T^2 and Q statistics for the ith driver as $\bar{T}^2_{(i)}$ and $\bar{Q}_{(i)}$.
9	End for

In Table 5.2, I represents the total number of participants and $y^{(i)}$ represents the awake RR intervals from the ith driver. In steps 1-3, for each participant the 8 HRV features presented in Section 5.1.3 were extracted for every 3 minute time window. Then these features were organized into a matrix $\tilde{X}^{(i)}$, where each row corresponds to a single time window and each column represents a feature. Next, in step 4, the individual matrices $\tilde{X}^{(i)}$ for each person were merged into one matrix $\tilde{X}$. During step 5, the merged matrix $\tilde{X}$ was preprocessed to obtain matrix X. This was done by standardizing each column of $\tilde{X}$, ensuring that each feature had a mean of 0 and a standard deviation of 1. This transformation is important because the features in $\tilde{X}$ may have different numerical scales. Without standardization, features with larger magnitudes would dominate the analysis, not because they are more important, but simply due to their scale. To perform this standardization, the mean and standard deviation of each column (feature) in $\tilde{X}$ were computed. Then, for every element x_{ij} in the matrix, where i indexes a time window and j indexes a feature, the mean of feature j (μ_j) was subtracted, and the result was divided by the standard deviation of that feature (σ_j). This is illustrated in Equation 5.12.

$$X_{ij} = \frac{\tilde{X}_{ij} - \mu_j}{\sigma_j} \tag{5.12}$$

In step 6, PCA is applied to X. In this step, the number of principal components R, which represents the dimensionality of the reduced feature space, must be carefully selected to ensure accurate drowsiness detection. Lastly, in steps 7-9, the T^2 and Q statistics are calculated and their control limits are defined. As HRV varies between individuals, the variation in T^2 and Q also differs per person. Therefore, control limits are determined separately for each driver.

5.1.6. Algorithm 2: Drowsy Driving Detection

Once the drowsiness detection model was prepared using Algorithm 1, Algorithm 2 was used to perform the actual drowsiness detection. As the proposed method in the paper is designed for real-time detection, but our analysis is performed offline after recording, the algorithm was slightly adapted to suit this setup. The algortihm is given in Table 5.3.

In Step 1, C represents the drivers state, which can either be Awake (A) or Drowsy (D). It is assumed that the participant starts in an alert state and becomes drowsy over time. The time counter variable τ is also initialized to zero. In Steps 2 till 4, the HRV features are extracted for each driver and organized into a feature matrix $X^{(i)}$, which is then preprocessed in the same way as in Algorithm 1. Step 6 involves calculating the $T^2[t]$ and $Q[t]$ statistics for each time window t. These statistics are then compared with the control limits $\overline{T^2}$ and $\overline{Q}$, obtained from Algorithm 1. If either $T^2[t] > \overline{T^2}$ or $Q[t] > \overline{Q}$ and the previous state was Awake ($C[t-1] = A$), or if both statistics are below their thresholds and the previous state was Drowsy ($C[t-1] = D$), the counter τ is incremented by one. If τ exceeds the predefined threshold $\overline{\tau}$, the drivers state is flipped (from Awake to Drowsy or vice versa), and τ is reset. This logic helps reduce false positives by ensuring that temporary spikes do not immediately trigger a state change. In this work, the value of $\overline{\tau}$ was set to 2, as it is important to detect drowsiness quickly without introducing long delays, in order to prioritize safety.

Table 5.3: Algorithm 2: Drowsy Detection

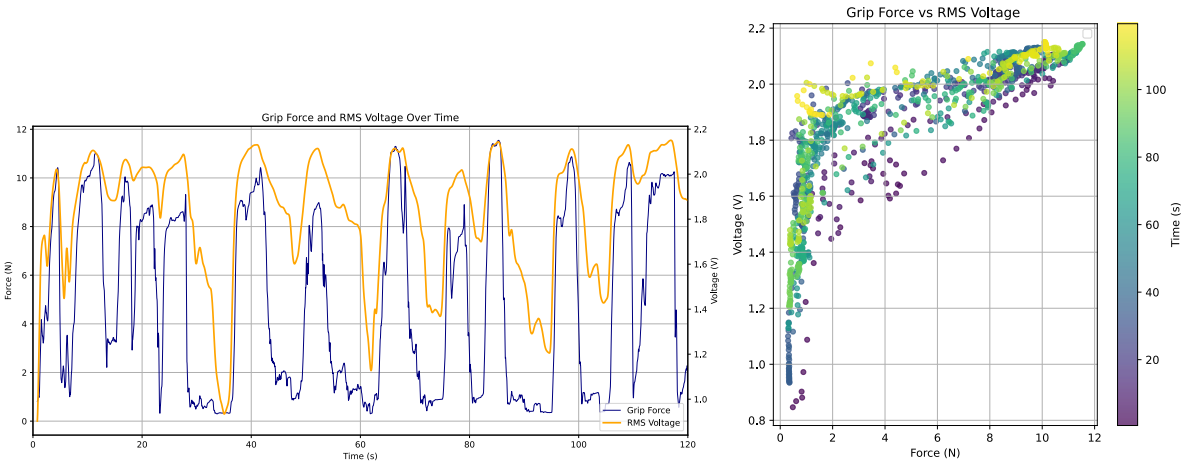
	Algorithm 2: Drowsy Driving Detection
1	Initialize driver state: $C \leftarrow A$, $\tau \leftarrow 0$
2	For each driver i:
3	Extract RR intervals $y^{(i)}$ and compute HRV features $X^{(i)}$
4	Preprocess to obtain feature matrix $X^{(i)}$
5	Compute $T^2[t]$ and $Q[t]$ for each time window t
6	For each time window t:
7	If $\big((T^2[t] > \overline{T^2} \vee Q[t] > \overline{Q}) \wedge C[t-1] = A\big)$ $\vee \big((T^2[t] \leq \overline{T^2} \wedge Q[t] \leq \overline{Q}) \wedge C[t-1] = D\big)$ then
8	$\tau \leftarrow \tau + 1$
9	Else: $\tau \leftarrow 0$
10	If $\tau \geq \overline{\tau}$ then
11	$C[t] \leftarrow \neg C[t-1]$, $\tau \leftarrow 0$
12	Else: $C[t] \leftarrow C[t-1]$

6

Experimental Results

6.1. Human-Antenna Sensor vs Pressure Sensor

To evaluate the reliability of the human-antenna grip sensing method, its signal output was compared with the output of the resistive force sensor, which serves as a reference or ground-truth measurement. Both sensors were recorded simultaneously during grip experiments under controlled conditions. The human-antenna sensor responds to changes in grip pressure via capacitive coupling, while the force sensor provides a direct measurement of applied force. Although the two sensors operate on fundamentally different physical principles, their outputs exhibit similar trends over time. In Figure 6.1a a 120 s recording of both sensor is shown. The participant changes grip over time from no grip to maximum grip. The voltage from our proposed sensor closely tracks the force sensor. However, the peak values remain consistent, whereas the minima vary every time. This could a limitation of our prototype design. It was intentionally designed to be small enough to fit entirely in any users hand, ensuring that hand size does not affect the measurements. However, this also introduces a limitation in measurement accuracy due to its compact size, the handle is pressed deeper into the palm during gripping. As a result, even after the grip is released, the sensor remains partially embedded in the palm, which may cause the signal to not full go to the minimum value and give an overestimation of force. The relationship between force and voltage is shown in Figure 6.1b, where a linear trend can be observed in the range of 1 N to 12 N. Despite noticeable deviations during each press and release cycle, the underlying correlation between the two signals is clearly present. These deviations are likely caused by hysteresis in the force sensor itself, which results in the sensor not following the exact same path during each cycle grip strength.

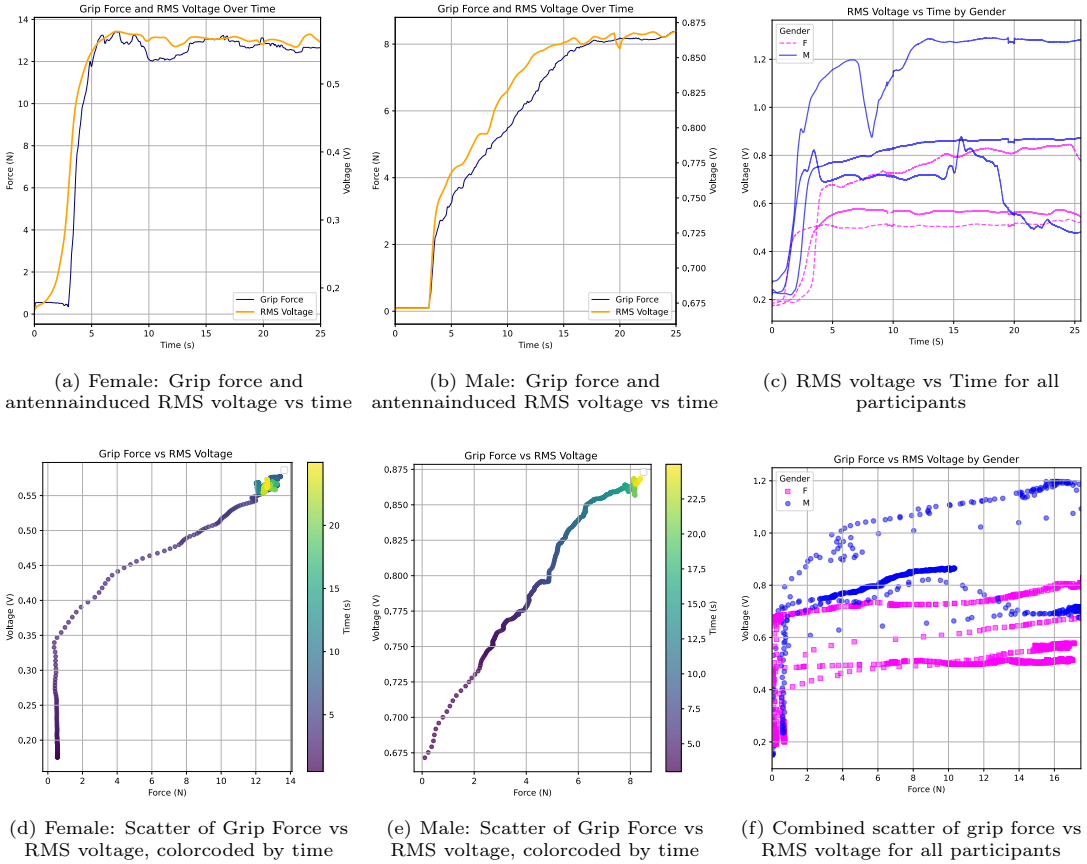


(a) RMS voltage and Force sensor compared in time

(b) Scatter plot of Voltage vs Force

Figure 6.1: Comparison of the proposed humanantenna sensor against the resistive force sensor during a variable increase in grip strength

Multiple measurements were conducted on participants with varying gender and body size. Figure 6.2 presents two measurements one from a male and one from a female subject each gradually increasing their grip strength. Although the term gradual is subjective and difficult to standardize across individuals, the results demonstrate that the sensor is capable of tracking both slow and fast changes in grip force. More importantly, the RMS voltage values differ significantly between genders: the male exhibited a range from approximately 675 mV to 875 mV while the female ranged from 180 mV to 575 mV. This suggests that although the antenna-based signal may not offer high absolute accuracy, it is sufficiently sensitive to capture dynamic variations in grip strength. As such, it shows promise as a non-invasive indicator for grip strength.



(a) Female: Grip force and antennainduced RMS voltage vs time

(b) Male: Grip force and antennainduced RMS voltage vs time

(c) RMS voltage vs Time for all participants

(d) Female: Scatter of Grip Force vs RMS voltage, colorcoded by time

(e) Male: Scatter of Grip Force vs RMS voltage, colorcoded by time

(f) Combined scatter of grip force vs RMS voltage for all participants

Figure 6.2: RMS voltage in time domain compared to force sensor, relationship between Force and RMS Voltage show from different gender gradually increasing there grip

Eight people participated in the experiment, comprising four males and four females. From figure 6.2c the rms voltage vs time is shown of all participants it can be observed that not all the participants are able to recreate a stable connection with sensor, this the limitation of our prototype that does not recreate good connection with the hand depending on the positioning. The force vs voltage relationship for all participants is shown in Figure 6.2f. The results indicate a significant discrepancy between male and female measurements, making it challenging to classify grip strength based solely on voltage values. In some cases, the maximum voltage measured for a female subject corresponds to the minimum for a male subject. This overlap highlights the difficulty of using uncalibrated voltage readings for subject-independent classification.

6.2. Algorithm 1: Drowsiness Detection Preparation

To reduce the dimensionality of the HRV feature set, Principal Component Analysis (PCA) was applied. The number of components R was chosen such that 95% of the total variance in the standardized features was retained. This threshold was chosen to keep most of the important information in the data while reducing complexity. This resulted in $R = 3$, as can be observed from Figure 6.3. This implies that three principal components were sufficient to represent most of the variability in the original data. This step ensures that the model remains efficient and less prone to overfitting, while still capturing important patterns in the HRV features.

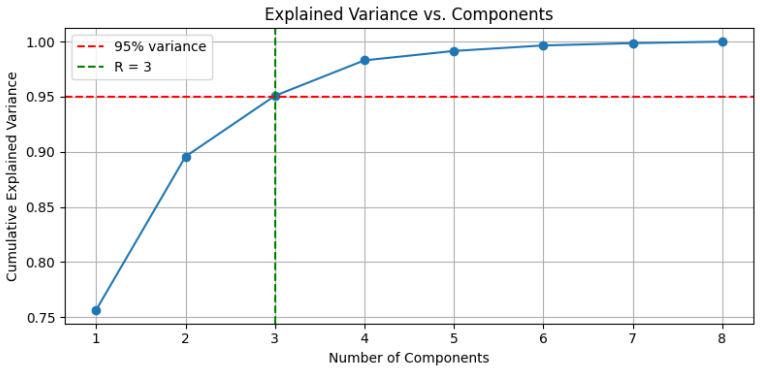
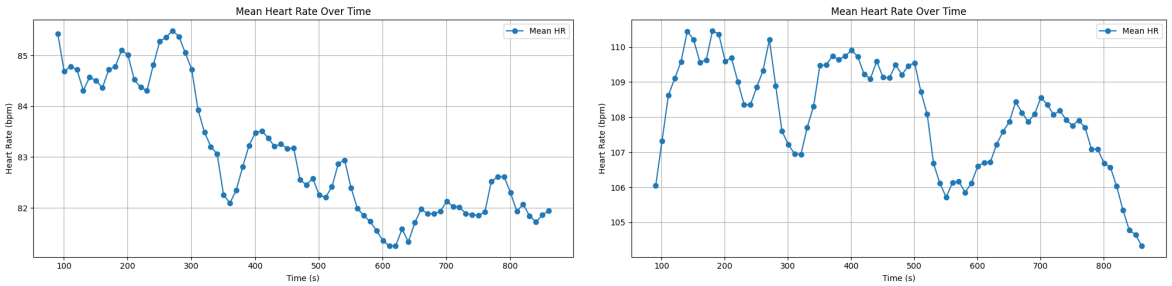


Figure 6.3: PCA R determination

Applying algorithm 1 resulted in the T^2 and the Q threshold statistics for every participant given in Table 6.1. These thresholds were calculated using their individual baseline data, as HRV varies a lot from person to person. It can be observed that both the T^2 and Q thresholds differ quite a bit across drivers, with driver 5 having the highest T^2 threshold and driver 4 having the highest Q threshold. A higher T^2 or Q threshold suggests that more variation in the HRV features is considered normal for that participant. So for participants with a higher threshold, the detection model will be less sensitive. In other words, the T^2 or Q value has to be much higher before the driver is flagged as drowsy. This could be because their baseline HRV was more variable to begin with. To illustrate this, the heart rate of Participant 5, who had relatively high thresholds, is compared with that of Participant 3, who had lower thresholds. As shown in the plots in Figure 6.4, Participant 5s heart rate is not only higher overall but also more variable. Since the T^2 and Q thresholds are based on the spread of a person's HRV features, more variability results in higher thresholds. In contrast, Participant 3 shows a lower and more stable heart rate, which explains the lower control limits.

Table 6.1: Hotellings T^2 and Q Control Limits for Each Driver (Algorithm 1)

Driver	T^2 Threshold	Q Threshold
Driver 1	1.878	0.430
Driver 2	7.977	0.466
Driver 3	2.310	0.254
Driver 4	4.770	4.505
Driver 5	15.858	1.362
Driver 6	8.509	1.212



(a) Driver 3 heartrate

(b) Driver 5 heartrate

Figure 6.4: Comparison of the heartrate

6.3. Algorithm 2: Drowsy Driving Detection

After preparing the individual models and control limits in Algorithm 1, Algorithm 2 was used to detect drowsiness over time for each participant. The goal was to observe how the driver's state changed based on the computed T^2 and Q statistics. These statistics were compared against the thresholds to determine when the driver transitioned from an awake to a drowsy state or vice versa. To visualize this, the drivers heart rate is shown along with shaded segments that indicate when drowsiness was detected. In Figure 6.5, the drowsy detection plot of participant 3 is plotted.

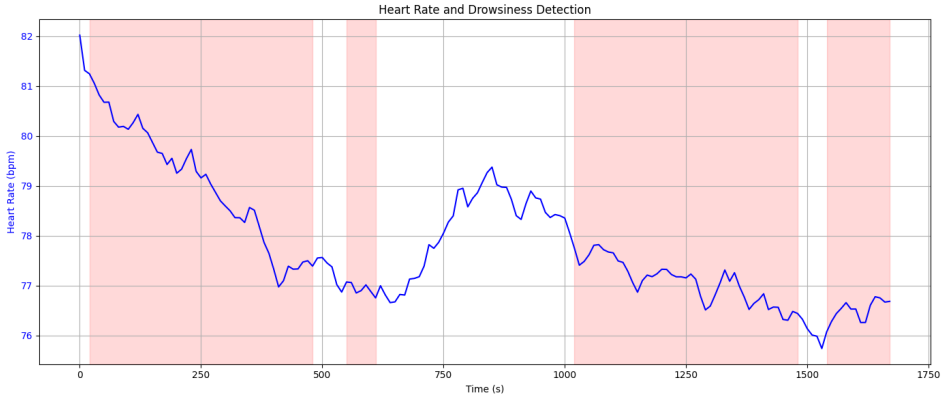
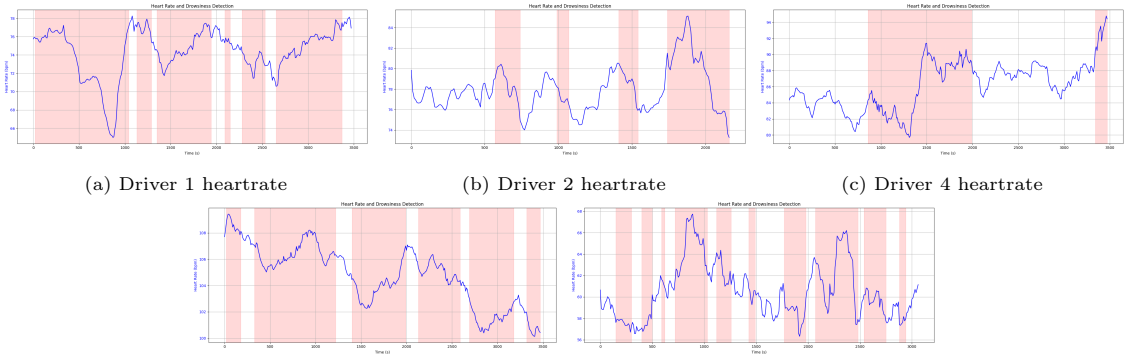


Figure 6.5: Drowsy Detection Participant 3

As time progresses, the heart rate gradually decreases, which is commonly associated with relaxation or reduced alertness. It can be observed from the plot that the parts where the heart rate drops are detected as drowsy, around 0 to 500 seconds, and again from 1000 to 1500 seconds. It is also noticeable that the section between 600 to 1000 seconds is detected as awake, which aligns with a rise in heart rate during that period. This suggests the model is sensitive to physiological changes that typically occur with drowsiness. However, some fluctuations and transitions also appear outside the red-shaded zones, showing that the heart rate alone is not always a perfect indicator of drowsiness, highlighting the importance of using multiple hrv features. The detection models of the remaining participants are given in Figure 6.6. It can be observed that in the plots of Driver 1 and Driver 5, more segments are detected as drowsy than awake. This might be due to false positives. Its important to note that the algorithm labels a segment as drowsy when it deviates from the drivers baseline. However, such deviations dont always indicate drowsiness. They could also result from stress or other emotional or physiological changes.



(a) Driver 1 heartrate (b) Driver 2 heartrate (c) Driver 4 heartrate

(d) Driver 5 heartrate (e) Driver 6 heartrate

Figure 6.6: Drowsiness Detection based on hrv

6.4. Grip rms and HRV drowsy detection

In addition to the HRV features, the root mean square (RMS) of the grip signal was extracted from the filtered grip data to observe changes during drowsy and awake states. Although grip RMS was not used for the detection itself, it was plotted alongside the drowsy segments to observe the behaviour between grip intensity and drowsiness. In Figure 6.7, the RMS signal is shown together with the drowsiness detection results from participant 3.

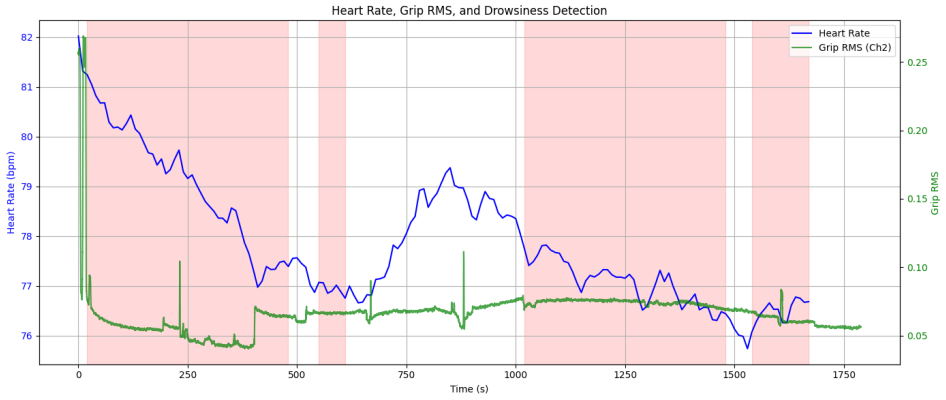


Figure 6.7: Drowsy Detection Participant 3 with RMS

The plot contains two axes: one showing the heart rate and the other showing the grip RMS. To make the changes in grip strength clearer, the RMS signal is zoomed in for both drowsy and awake segments, as shown in Figure 6.8. It can be observed that the RMS decreases during drowsy periods, which indicates a weaker grip. On the other hand, the RMS increases during awake segments, suggesting that the participant is tightening their grip.

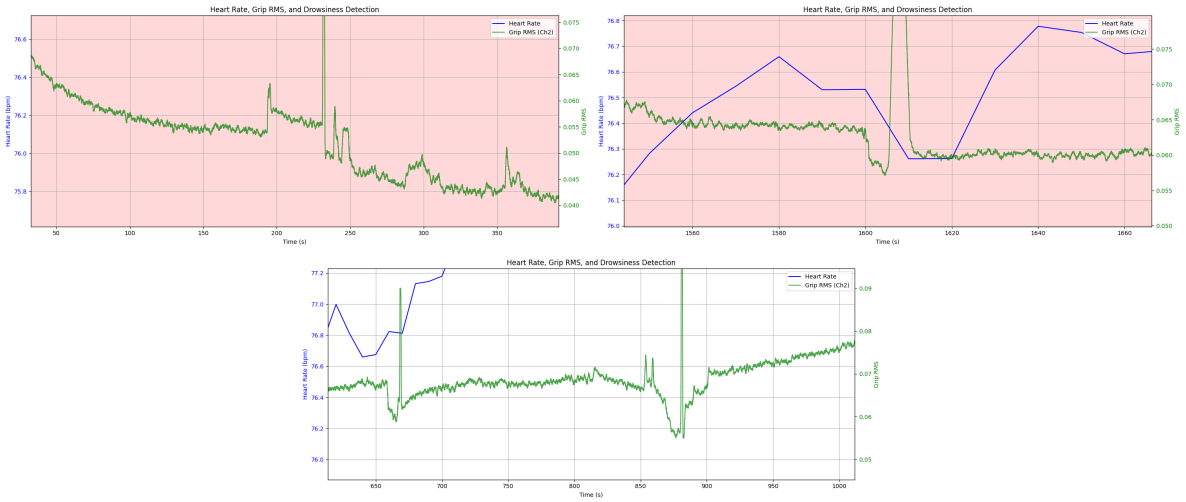


Figure 6.8: RMS during drowsy and awake data

7

Conclusion and future works

7.1. Conclusion

Drowsy driving remains a critical yet often underestimated factor contributing to road accidents worldwide. This project has explored a novel, non-intrusive approach to detecting driver drowsiness by using the Human antenna effect to measure grip strength and validate it with the heart rate by the use of an ECG. Through this research, we have demonstrated a proof of concept of the proposed sensor to determine grip strength, Although promising, it was not possible to classify grip strength using voltage threshold hold. There was a significant discrepancy between men and women. This could be corrected by calibrating the sensor per person, which contradicts our primary objective of not disrupting the drivers experience. Further research is needed to improve sensor performance. For the drowsiness detection part, the grip strength features were not able to be used to clasify drowsines due to time constraints. However, it was observed that the grip rms decreases as drowsiness is detected by the hrv. This marks the potential for early, passive detection of drowsiness that prioritizes driver comfort and privacy.

7.2. Future works

The sensor model requires further research to develop a more accurate model and gain a deeper understanding of the sensors dynamics. While this thesis focused on capacitive coupling, it is crucial to consider various coupling methods between the human and the sensor. Additionally, a hybrid approach combining isolated and bare wire could be implemented in the steering mechanism. Using insulated wire to determine grip strength via capacitive coupling, while the bare wire could function as an electrode to measure heart rate variability, assuming the individual is driving with both hands.

7.2.1. Data collection

Due to limited time, the dataset collected in this study is relatively small, which limits the potential accuracy of the analysis. For future work, it will be essential to gather a larger and more diverse dataset that better represents various driver behaviors and conditions. This expanded data will provide a stronger foundation for more advanced analysis techniques. Also, the grip calibration of the participants can be used as grip features for the dataset, as those features characterize the light, medium and maximum grip of each individual.

7.2.2. Machine learning

Although machine learning was not applied in this project, it presents a promising direction for future research. A Supervised machine learning model could be trained on an expanded dataset to accurately distinguish between relaxed but alert states and true drowsiness by detecting patterns in grip strength data. Using these techniques could significantly improve the reliability and robustness of drowsiness detection systems.

7.2.3. MSPC

Another way to classify drowsiness as a future work based on the grip is to apply the MSPC algorithms on the grip features and to observe if the drowsy sections of the grip detection model overlap with the hrv based MSPC detection model. The accuracy of the models may increase as the dataset size increases. Also, having longer baseline recordings might also increase the accuracy, as you have a more reliable data segment to compare against, which can reduce false positives.

Bibliography

[1] World Health Organization, "Road traffic injuries," https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries, 2023, [Accessed: 2025-05-02].

[2] SWOV - Instituut voor Wetenschappelijk Onderzoek Verkeersveiligheid, "Vermoeidheid," https://swov.nl/en/fact-sheet/fatigue, 2019, [Accessed: 2025-05-02].

[3] M. Gonçalves and et al, "Sleepiness at the wheel across europe: A survey of 19 countries," Journal of Sleep Research, vol. 24, no. 3, pp. 242–253, 2015.

[4] SWOV - Institute for Road Safety Research, "Road crash costs," 2024, https://swov.nl/en/fact/road-crash-costs-1-what-are-costs-road-crashes [Accessed: 2025-05-02].

[5] National Center for Biotechnology Information (NCBI), "Medgen: Narcolepsy [internet]," https://www.ncbi.nlm.nih.gov/medgen/4390, 2025, [Accessed: 2025-05-02].

[6] J. F. Pagel, "Excessive daytime sleepiness," American Family Physician, vol. 79, no. 5, 2009, [Accessed: 2025-05-02]. [Online]. Available: https://www.aafp.org/pubs/afp/issues/2009/0301/p391.pdf

[7] H. Pan and et al, "Exploring the effect of driver drowsiness on takeover performance during automated driving: An updated literature review," Accident Analysis Prevention, vol. 216, p. 108023, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0001457525001095

[8] M. Ngxande, J.-R. Tapamo, and M. Burke, "Driver drowsiness detection using behavioral measures and machine learning techniques: A review of state-of-art techniques," in 2017 Pattern Recognition Association of South Africa and Robotics and Mechatronics (PRASA-RobMech), 2017, pp. 156–161.

[9] E. Perkins and et al, "Challenges of driver drowsiness prediction: The remaining steps to implementation," IEEE Transactions on Intelligent Vehicles, vol. 8, no. 2, pp. 1319–1338, 2021.

[10] S. Siddharth and M. M. Trivedi, "On assessing driver awareness of situational criticalities: Multimodal bio-sensing and vision-based analysis, evaluations, and insights," Brain Sciences, vol. 10, no. 1, 2020.

[11] Mayo Clinic Staff, "Eeg (electroencephalogram)," https://www.mayoclinic.org/tests-procedures/eeg/about/pac-20393875, [Accessed 02-05-2025].

[12] ——, "Electrocardiogram (ecg or ekg)," https://www.mayoclinic.org/tests-procedures/eeg/about/pac-20393875, [Accessed 02-05-2025].

[13] F. B. Reguig, "Photoplethysmogram signal analysis for detecting vital physiological parameters: An evaluating study," in 2016 International Symposium on Signal, Image, Video and Communications (ISIVC), 2016, pp. 167–173.

[14] C. E. Yekta Said Can, Bert Arnrich, "Stress detection in daily life scenarios using smart phones and wearable sensors: A survey," Journal of Biomedical Informatics, vol. 92, p. 103139, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1532046419300577

[15] Volvo Cars, "Volvo s60 support driver alert control," https://www.volvocars.com/mt/support/car/s60/article/f57bafffba0c0468c0a8015174f226d8/, 2024, [Accessed: Apr. 29, 2025].

[16] Tesla, "Model 3 owners manual," https://www.tesla.com/ownersmanual/model3/en_eu/GUID-65BF21B8-50C5-4FA5-86A4-DA363DCD0484.html, 2024, [Accessed: Apr. 29, 2025].

[17] Mercedes-Benz USA, "Function of attention assist with microsleep detection eqs sedan manual," https://www.mbusa.com/en/owners/manuals/eqs-sedan-2022-03-v297-mbux/attention-assist/function-of-attention-assist-with-microsleep-detection, 2022, [Accessed: Apr. 29, 2025].

[18] Tesla, "Tesla vision," https://www.tesla.com/support/transitioning-tesla-vision, 2025.

[19] G. S. Maximous and H. A. Bastawrous, "Driver drowsiness detection based on humantenna effect for automotive safety systems," in 2020 IEEE 9th Global Conference on Consumer Electronics (GCCE), 2020, pp. 391–392.

[20] K. Fujiwara and et al, "Heart rate variability-based driver drowsiness detection and its validation with eeg," IEEE Transactions on Biomedical Engineering, vol. 66, no. 6, pp. 1769–1778, 2019.

[21] M. H. Kryger, T. Roth, and C. A. Goldstein, Principles and Practice of Sleep Medicine, 7th ed., 2022.

[22] T. Åkerstedt and M. Gillberg, "Subjective and objective sleepiness in the active individual," International Journal of Neuroscience, vol. 52, no. 1-2, pp. 29–37, 1990. [Online]. Available: https://doi.org/10.3109/00207459008994241

[23] J. V. Forrester and et al, "Chapter 5 - physiology of vision and the visual system," in The Eye (Fourth Edition), fourth edition ed., J. V. Forrester, A. D. Dick, P. G. McMenamin, F. Roberts, and E. Pearlman, Eds. W.B. Saunders, 2016, pp. 269–337.e2. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780702055546000058

[24] T. Åkerstedt and et al, "Impaired alertness and performance driving home from the night shift: a driving simulator study," Journal of sleep research, vol. 1, no. 1, pp. 17–20, 2005. [Online]. Available: https://doi.org/10.1111/j.1365-2869.2004.00437.x

[25] J. Trinder and et al, "Impaired alertness and performance driving home from the night shift: a driving simulator study," Journal of sleep research, vol. 10, no. 4, p. 253264, 2001. [Online]. Available: https://doi.org/10.1046/j.1365-2869.2001.00263.x

[26] A. J. Camm and M. Malik, "Guidelines heart rate variability - standards of measurement, physiological interpretation, and clinical use," Eur. Heart J., vol. 115, no. 5, pp. 354–381, 1996.

[27] F. Versace and et all, "Heart rate variability during sleep as a function of the sleep cycle," Biological Psychology, vol. 63, no. 2, pp. 146–162, 2003.

[28] F. T. Ulaby and U. Ravaioli, Fundamentals of Applied Electromagnetics, 7th ed. Boston, MA: Pearson Education, 2015.

[29] G. Cohn and et al, "Your noise is my command: Sensing gestures using the body as an antenna," in Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ser. CHI '11. New York, NY, USA: ACM, 2011, pp. 791–800. [Online]. Available: http://doi.acm.org/10.1145/1978942.1979058

[30] ——, "Humantenna: Using the body as an antenna for real-time whole-body interaction," in Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ser. CHI '12. New York, NY, USA: ACM, 2012, pp. 1901–1910. [Online]. Available: http://doi.acm.org/10.1145/2207676.2208330

[31] L. F. Chaparro and A. Akan, "Introduction to the design of discrete filters," in Signals and Systems Using MATLAB, 3rd ed. Academic Press, 2019, ch. 12, pp. 721–802.

[32] Q. Qin and et all, "An adaptive and time-efficient ecg r-peak detection algorithm," Journal of Healthcare Engineering, vol. 2017, 2017.

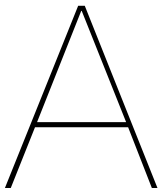
[33] G. Li and et all, "Detection of driver drowsiness using wavelet analysis of heart rate variability and a support vector machine classifier," Sensors, vol. 13, no. 12, dec 2013. [Online]. Available: https://doi.org/10.3390/s131216494

[34] D. Ewing, J. Neilson, and P. Travis, "New method for assessing cardiac parasympathetic activity using 24 hour electrocardiograms," British Heart Journal, vol. 52, no. 4, pp. 396–402, oct 1984.

[35] F. Shaffer and J. P. Ginsberg, "An overview of heart rate variability metrics and norms," Frontiers in Public Health, vol. 5, p. 258, 2017.

[36] M. Kano and et al, "A new multivariate statistical process monitoring method using principal component analysis," Computers Chemical Engineering, vol. 25, no. 7, pp. 1103–1113, 2001. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0098135401006834

Appendices



Forms

A.1. Consent form

CONSENT FORM

Drowsiness detection

The purpose of this research is to explore how grip strength on a steering wheel relates to drowsiness. When you fall asleep, your heart rate and grip strength change. By measuring both, we aim to get a better understanding of the physical signs of drowsiness while driving.
You will drive in a driver simulator with a steering wheel. The simulation is designed to mimic real driving, but in a safe and controlled environment.

Procedure
I confirm that the research project has been explained to me and I have had the opportunity to ask any questions I may have.

Voluntary Participation and withdrawal
I understand that my participation is voluntary and I am free to withdraw at any time without providing any reason and without facing any negative consequences. I also understand that I am not required to answer any questions I do not wish to, and I may decline to respond to specific questions without penalty.

Confidentiality
I understand that all my responses will be kept strictly confidential. I give permission for members of the research team to have access to my anonymised responses. I also understand that my name will not be associated with any research materials and that I will not be identified or identifiable in the report or reports that result from the research.
Additionally, I consent to my anonymized data being used for future research or reference.

Risks
I do understand the potential risk of hairloss where the electrodes are attached and the risk of being in an induced stressy situation.

Participation
By signing this form, I confirm my willingness to take part in the research project described above.

Name of participant:
Date:

Signature of participant

Contact person: Mai Anh Nguyen e-mail: M.A.Nguyen@student.tudelft.nl

Figure A.1: Consent form

A.2. Questionnaire

QUESTIONAIRE

Question 1 Select your gender
- o Female
- o Male
- o Other

Question 2 Do you have cardiovascular disease?
- o Yes
- o No

Question 3 Do you have arrhythmia? (Abnormal heartbeat)
- o Yes
- o No

Question 4 Do you have epilepsy?
- o Yes
- o No

Question 5 Do you have hypertension?
- o Yes
- o No

Question 6 Do you have diabetes?
- o Yes
- o No

Question 7 Do you have a sleep disorder?
- o Yes
- o No

Question 8 Do you have any health concerns?
- o Yes
- o No

Question 9 Do you smoke?
- o Yes
- o No

Question 10 Do you have a driver's licence?
- o Yes
- o No

Question 11 How many hours do you sleep on average?
Please round up.

_____

Question 12 How many hours did you sleep last night?

_____

Question 13 Did you consume alcohol yesterday?

- o Yes
- o No

Figure A.2: Questionnaire

B

Code

B.1. Source Code listing

B.1.1. Data Acquisition code used on the esp32

```
#include <SPI.h> // use the SPI library to facilatete the comumation

#define CS_PIN       5 // chip select pin
#define NUM_CHANNELS 6 // number of channels
#define SAMPLE_RATE  1000  //  sampling rate in Hz

hw_timer_t *timer = nullptr; // create a pointer to the internal timer struct and intializing to 0
portMUX_TYPE timerMux = portMUX_INITIALIZER_UNLOCKED;// insure that the sampleready is only accesed by 1 one function at a time.
volatile bool sampleReady = false; // flag for the time interupt
uint16_t latest[NUM_CHANNELS];// buffer that hold all the sampled values

//add this function on the instruction memory to reduce the latency and jitter
void IRAM_ATTR onTimer() {
  portENTER_CRITICAL_ISR(&timerMux);
  sampleReady = true;
  portEXIT_CRITICAL_ISR(&timerMux);
}

//SPI commands send to the ADC
uint16_t readMCP3208(uint8_t ch) {
  uint8_t cmd = 0b00000110 | ((ch & 0b100) >> 2);//
  uint8_t msb = (ch & 0b011) << 6;

  SPI.beginTransaction(SPISettings(2000000, MSBFIRST, SPI_MODE0));//start SPI comunication @ 2 MHz clock speed
  digitalWrite(CS_PIN, LOW); // set CS 0 to start the comunication between the adc and esp32
  SPI.transfer(cmd);// send first byte
  uint8_t h = SPI.transfer(msb) & 0x0F;// send second byte
  uint8_t l = SPI.transfer(0x00);
  digitalWrite(CS_PIN, HIGH);
  SPI.endTransaction();
  return (h << 8) | l; //combine high and low to get the full 12bit output
}

void setup() {
  Serial.begin(921600);//start serial comunation between pc and esp32
```

```
  while (!Serial);

  SPI.begin(18, 19, 23, CS_PIN);// declare pin out: MOSI=18, MISO=19,SCK=23
   ↪ ,CS_PIN is define in the beginning
  pinMode(CS_PIN, OUTPUT);
  digitalWrite(CS_PIN, HIGH);

  timer = timerBegin(1000000);  //set the timers clk speed at 1 micro seconds or
   ↪ 1Mhz

  timerAttachInterrupt(timer, onTimer); //call onTimer when the 1ms is counted

  timerAlarm(timer, 1000000 / SAMPLE_RATE, true, 0);//set the time for the timer
   ↪ (clock rate, interupt time,repeat ,0)

  timerStart(timer);// start the timer

  Serial.println("timestamp_us,ch0,ch1,ch2,ch3,ch4,ch5");
}

void loop() {
  //wait till sample is ready and change sample to false again
  if (!sampleReady) return;
  portENTER_CRITICAL(&timerMux);
    sampleReady = false;
  portEXIT_CRITICAL(&timerMux);

  uint32_t t = micros();//records the time in micro seconds at which timw the
   ↪ sample is being recorded
  for (uint8_t i = 0; i < NUM_CHANNELS; i++) {
    latest[i] = readMCP3208(i);//read all the channels
  }

  // print print the time before we started reading and all the channels via
   ↪ serial
  Serial.print(t);
  for (uint8_t i = 0; i < NUM_CHANNELS; i++) {
    Serial.print(',');
    Serial.print(latest[i]);
  }
  Serial.println();
}
```

B.1.2. Python Code for Retrieving Data from the ESP32 and Oscilloscope

```
import threading
import time
import csv
import signal
import numpy as np
import dwfpy as dwf
import datetime
import serial
# Adapted from "analog_in_logger.py" by Marius Greuel (2024)
# Original source:
 ↪ https://github.com/mariusgreuel/dwfpy/blob/main/examples/analog_in_logger.py
# Modifications for ESP32 data logging integration.
# === Parameters ===
```

```
Record_time = 30
SAMPLE_RATE = 1000  # Hz
BUFFER_SIZE = 10000   # Samples
READ_INTERVAL = 1*(BUFFER_SIZE/SAMPLE_RATE)  # 10 Seconds
CSV_FILE = "recf2.csv"

# === Thread control flag ===
running = True

# === Shared queue to pass data from thread to logger ===
osc_queue = []
esp_queue = []

# === Oscilloscope Thread Function ===
def record_oscilloscope():
    with dwf.Device() as device:
        print(f"Connected to: {device.name} ({device.serial_number})")
        scope = device.analog_input

        for ch in [0, 1]:
            scope[ch].setup(range=5)
            scope[ch].setup(range=2)

        scope.sample_rate = SAMPLE_RATE
        scope.buffer_size = BUFFER_SIZE
        scope.scan_shift(sample_rate=SAMPLE_RATE, buffer_size=BUFFER_SIZE,
         ↪  configure=True, start=True)

        # Initialize time tracking
        dt = 1 / SAMPLE_RATE
        sample_index = 0
        start_time = time.perf_counter()

        while running:
            time.sleep(READ_INTERVAL)
            scope.read_status(read_data=True)
            ch1 = np.array(scope[0].get_data())
            ch2 = np.array(scope[1].get_data())

            # Compute timestamps from sample index
            ts_start = start_time + sample_index * dt
            osc_queue.append((ts_start, ch1.tolist(), ch2.tolist()))

            sample_index += len(ch1)


def record_esp32_serial(port="COM5", baud=921600):
    try:
        ser = serial.Serial(port, baud, timeout=0.5)
        print(f"Connected to ESP32 on {port} @ {baud} baud.")
    except serial.SerialException as e:
        print(f"[ERROR] Could not open serial port: {e}")
        return

    try:
        while running:
            try:
                line = ser.readline().decode(errors='ignore').strip()
                if line:
                    try:
```

```
                        parts = [int(x.strip()) for x in line.split(',')]
                        if len(parts) == 2:
                            esp_time = parts[0] / 1_000_000  # µs to sec
                            val = parts[1]
                            host_time = time.perf_counter()
                            esp_queue.append((host_time, [esp_time, val]))
                        else:
                            print(f"[ESP PARSE WARNING] Invalid line: {line}")
                    except Exception as e:
                        print(f"[ESP PARSE ERROR] Line: {line} → {e}")
            except Exception as read_err:
                print(f"[ESP32 Read Error] {read_err}")
                break
    finally:
        ser.close()
        print("ESP32 serial closed.")


# === Signal handler for Ctrl+C ===
# Define stop_all so it works for both signal and timer
def stop_all(sig=None, frame=None):
    global running
    print("Stopping recording...")
    running = False

# Register Ctrl+C handler
signal.signal(signal.SIGINT, stop_all)

# Start auto-stop timer
stop_timer = threading.Timer(Record_time, stop_all)
stop_timer.start()

signal.signal(signal.SIGINT, stop_all)

# === Data Saver (Main Thread) ===
def gather_and_save():
    all_rows = []
    osc_sample_rate = SAMPLE_RATE  # same as used in record_oscilloscope

    print("Logging to CSV... Press Ctrl+C to stop.")

    empty_loops = 0
    max_empty_loops = 40  # ~2 seconds idle

    while True:
        wrote_data = False

        # === Handle oscilloscope batches ===
        while osc_queue:
            ts, ch1, ch2 = osc_queue.pop(0)
            dt = 1 / SAMPLE_RATE
            for i in range(len(ch1)):
                t_sample = ts + i * dt
                all_rows.append([t_sample, ch1[i], ch2[i], None, None])
            wrote_data = True


        # === Handle ESP32 real-time samples ===
        while esp_queue:
            host_ts, val = esp_queue.pop(0)
```

```
            esp_time, sensor_val = val
            # You can choose to use `host_ts` or `esp_time`
            all_rows.append([host_ts, None, None, esp_time, sensor_val])

            wrote_data = True

        if not wrote_data:
            empty_loops += 1
            if not running and empty_loops > max_empty_loops:
                print("Queues idle and stopped. Exiting logger.")
                break
        else:
            empty_loops = 0

        time.sleep(0.005)

    # === Sort by timestamp ===
    all_rows.sort(key=lambda row: row[0])

    # === Write to CSV ===
    with open(CSV_FILE, "w", newline='') as f:
        writer = csv.writer(f)
        writer.writerow(["timestamp", "osc_ch1", "osc_ch2", "esp_timestamp",
         ↪  "esp_value"])
        writer.writerows(all_rows)

    print("Recording complete.")

osc_thread = threading.Thread(target=record_oscilloscope)
esp_thread = threading.Thread(target=record_esp32_serial)

# make them daemon threads so they won't hang the process if something goes wrong
osc_thread.daemon = True
esp_thread.daemon = True

osc_thread.start()
esp_thread.start()

# wait for the auto-stop timer to fire, which calls stop_all()
stop_timer.join()                # blocks until Record_time elapses

# now both threads will naturally exit their loops (running==False)
print("Waiting for acquisition threads to finish…")
osc_thread.join(timeout=1.0)     # give them up to 1s to wrap up
esp_thread.join(timeout=1.0)

# now safely gather whatever's left in the queues and write the CSV
gather_and_save()

print("All threads joined. Program ended.")
```

B.1.3. Python Code for Algorithm 1

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt
```

```
6
7   from scipy.signal import iirnotch, filtfilt, butter
8
9   import os
10
11  from ecgdetectors import Detectors
12
13  import math
14
15  from hrv import HRV
16
17  import neurokit2 as nk
18
19  from sklearn.preprocessing import StandardScaler
20
21  from sklearn.decomposition import PCA
22
23  import scipy.stats as stats
24
25  import seaborn as sns
26
27  from hrvanalysis import remove_outliers, remove_ectopic_beats,
     ↪  interpolate_nan_values
28
29  from hrvanalysis import get_time_domain_features, get_frequency_domain_features
30
31  import joblib
32
33
34
35
36
37  def process_ecg_to_hrv_features(filepath):
38
39
40
41      # === Load and Clean ===
42
43      df = pd.read_csv(filepath).iloc[1:]
44
45      esp_df = df.dropna(subset=["esp_value"]).copy()
46
47
48
49      fs = 1 / np.mean(np.diff(esp_df['timestamp'].values))
50
51      print(f"Sampling rate (fs) is {fs:.2f} Hz")
52
53
54
55      # === Filters ===
56
57      def notch_filter(data, notch_freq=50.0, fs=fs, Q=30):
58
59          b, a = iirnotch(notch_freq, Q, fs)
60
61          return filtfilt(b, a, data)
62
63
64
```

```
65      def bandpass_filter(data, lowcut=0.5, highcut=20, fs=fs, order=2):
66
67          nyq = 0.5 * fs
68
69          low = lowcut / nyq
70
71          high = highcut / nyq
72
73          b, a = butter(order, [low, high], btype='band')
74
75          return filtfilt(b, a, data)
76
77
78
79      def smooth(data, window_size=5):
80
81          return np.convolve(data, np.ones(window_size)/window_size, mode='same')
82
83
84
85      # === Split into windows ===
86
87      def split_by_time(df, window_sec):
88
89          windows = []
90
91          start_time = df['timestamp'].iloc[0]
92
93          end_time = math.ceil(df['timestamp'].iloc[-1])
94
95          current_start = start_time
96
97          while current_start + window_sec <= end_time:
98
99              current_end = current_start + window_sec
100
101             window = df[(df['timestamp'] >= current_start) & (df['timestamp'] <
                ↪  current_end)].reset_index(drop=True)
102
103             if not window.empty:
104
105                 windows.append(window)
106
107             current_start = current_end
108
109         return windows
110
111
112
113     def split_by_time_test(df, window_sec, step_sec):
114
115         windows = []
116
117         start_time = df['timestamp'].iloc[0]
118
119         end_time = math.ceil(df['timestamp'].iloc[-1])
120
121         current_start = start_time
122
123
```

```
124
125         while current_start + window_sec <= end_time:
126
127             current_end = current_start + window_sec
128
129             window = df[(df['timestamp'] >= current_start) & (df['timestamp'] <
                ↪  current_end)].reset_index(drop=True)
130
131             if not window.empty:
132
133                 windows.append(window)
134
135             current_start += step_sec  # <-- This creates the sliding behavior
136
137
138
139         return windows
140
141
142
143     # === R-peak Detection ===
144
145     def ecg_peaks(x, sampling_rate):
146
147         detectors = Detectors(sampling_rate)
148
149         search_interval = 150  # in samples
150
151
152
153         # Step 1: Pan-Tompkins Detection
154
155         R = np.asarray(detectors.pan_tompkins_detector(x))
156
157
158
159         # Step 2: Refine to local max
160
161         for j in range(R.size):
162
163             left = max(R[j] - search_interval, 0)
164
165             right = min(R[j] + search_interval, len(x))
166
167             local_peak = np.argmax(x[left:right])
168
169             R[j] = local_peak + left
170
171
172
173         # Step 3: Remove duplicates
174
175         R = np.unique(R)
176
177
178
179         # Step 4: Remove low peaks
180
181         peak_heights = x[R]
182
```

```
        threshold = np.median(peak_heights) * 0.5

        R = R[peak_heights > threshold]


        # Step 5: Remove peaks too close together (< 300ms)

        min_interval = int(0.3 * sampling_rate)

        final_peaks = [R[0]]

        for i in range(1, len(R)):

            if R[i] - final_peaks[-1] >= min_interval:

                final_peaks.append(R[i])

        return np.array(final_peaks)


    # === Process Signal ===

    esp_df['esp_scaled'] = (esp_df['esp_value'].astype(float) * 5) / 4096

    esp_df['ecg_filtered'] =
     ↪ bandpass_filter(esp_df['esp_scaled'].values.astype(float), fs=fs)

    signal = np.array(esp_df['ecg_filtered'], dtype=np.float32)

    esp_df['smoothed'] = smooth(signal, window_size=5)


    # windows = split_by_time(esp_df, 180)

    windows = split_by_time_test(esp_df, 180, 10)

    print(f"Number of 180s windows: {len(windows)}")


    # === Detect R-peaks ===

    r_peaks = ecg_peaks(esp_df['smoothed'].values, sampling_rate=fs)

    r_peak_times = esp_df['timestamp'].values[r_peaks]

    r_peak_amplitudes = esp_df['smoothed'].values[r_peaks]


    # === Segment Peaks by Time ===

    window_size = 180

    start_time = esp_df['timestamp'].iloc[0]

    end_time = esp_df['timestamp'].iloc[-1]
```

```
242
243
244
245     r_peaks_list, r_peak_times_list, r_peak_amplitudes_list = [], [], []
246
247     current_start = start_time
248
249     step_size = 10
250
251
252
253
254
255     while current_start < end_time:
256
257         current_end = min(current_start + window_size, end_time)
258
259         mask = (r_peak_times >= current_start) & (r_peak_times < current_end)
260
261
262
263         window_r_peak_times = r_peak_times[mask]
264
265         window_r_peak_amplitudes = r_peak_amplitudes[mask]
266
267         window_r_peak_indices = r_peaks[mask]
268
269
270
271         if len(window_r_peak_times) > 1:
272
273             duration = window_r_peak_times[-1] - window_r_peak_times[0]
274
275             if duration >= 120:
276
277                 r_peaks_list.append(window_r_peak_indices)
278
279                 r_peak_times_list.append(window_r_peak_times)
280
281                 r_peak_amplitudes_list.append(window_r_peak_amplitudes)
282
283         current_start += step_size
284
285
286
287     # === RR Interval Processing ===
288
289     rr_intervals_list_ms = [np.diff(times) * 1000 for times in r_peak_times_list]
290
291
292
293     nn_intervals_list_ms = []
294
295     for rr in rr_intervals_list_ms:
296
297         rr_no_outliers = remove_outliers(rr_intervals=rr, low_rri=300,
            ↪  high_rri=2000)
298
299         rr_interp = interpolate_nan_values(rr_intervals=rr_no_outliers)
300
```

```
        nn = remove_ectopic_beats(rr_intervals=rr_interp, method="malik")

        nn_interpolated = interpolate_nan_values(rr_intervals=nn)

        nn_intervals_list_ms.append(nn_interpolated)


    # === HRV Feature Extraction ===

    mean_nni_list, rmssd_list, sdnn_list = [], [], []

    nni_50_list, mean_hr_list = [], []

    LF_list, HF_list, LF_HF_ratio_list, total_power_list = [], [], [], []


    for nn in nn_intervals_list_ms:

        if len(nn) > 2:

            t_features = get_time_domain_features(nn)

            f_features = get_frequency_domain_features(nn)


            mean_nni_list.append(t_features['mean_nni'])

            rmssd_list.append(t_features['rmssd'])

            sdnn_list.append(t_features['sdnn'])

            nni_50_list.append(t_features['nni_50'])

            mean_hr_list.append(t_features['mean_hr'])


            LF_list.append(f_features['lf'])

            HF_list.append(f_features['hf'])

            LF_HF_ratio_list.append(f_features['lf_hf_ratio'])

            total_power_list.append(f_features['total_power'])

        else:

            mean_nni_list.append(np.nan)

            rmssd_list.append(np.nan)

            sdnn_list.append(np.nan)

            nni_50_list.append(np.nan)

            mean_hr_list.append(np.nan)
```

```
            LF_list.append(np.nan)

            HF_list.append(np.nan)

            LF_HF_ratio_list.append(np.nan)

            total_power_list.append(np.nan)



    # === Plot ECG Signal with R-peaks ===

    plt.figure(figsize=(12, 5))

    plt.plot(windows[0]['timestamp'], windows[0]['smoothed'], color='blue',
     ↪  label='Smooth ECG')

    plt.plot(r_peak_times_list[0], r_peak_amplitudes_list[0], 'ro',
     ↪  label='R-peaks')

    plt.title("ECG Signal with R-Peaks")

    plt.xlabel("Time (s)")

    plt.ylabel("Voltage (V)")

    plt.legend()

    plt.grid(True)

    plt.tight_layout()

    #plt.show()



    # === Plot HRV Over Time ===

    # num_windows = len(windows)

    # window_times = np.array([start_time + (i + 0.5) * window_size for i in
     ↪  range(num_windows)])

    window_times = np.array([start_time + i * step_size + window_size / 2 for i
     ↪  in range(len(mean_hr_list))])




    plt.figure(figsize=(10, 5))

    plt.plot(window_times, mean_hr_list, marker='o', label='Mean HR')

    plt.xlabel('Time (s)')

    plt.ylabel('Heart Rate (bpm)')

    plt.title('Mean Heart Rate Over Time')

```

```
    plt.legend()

    plt.grid(True)

    plt.tight_layout()

    plt.show()




    # Algorithm 1:

    # make matrix X with the features for the person

    X_i = np.column_stack((mean_nni_list, sdnn_list, rmssd_list,
     ↪  total_power_list, nni_50_list, LF_list, HF_list, LF_HF_ratio_list))

    print(f"X_i is {X_i}")



    return X_i


# To save data for algorithm 2

model_dir =
 ↪  r"C:\Users\harim\Desktop\grip_code\BAP25\Measurement_code\models_testoutliers"

os.makedirs(model_dir, exist_ok=True)



# --- Step 1: Filepaths ---

path = r"C:\Users\harim\Desktop\grip_code\BAP25\Measurement_code\ecg_baseline"



files = {

    "sanne": "ecg_baseline_sanne.csv",

    "david": "ecg_baseline_david.csv",

    "harim": "ecg_baseline_harim.csv",

    "maianh": "ecg_baseline_maianh.csv",

    "peter": "ecg_baseline_peter.csv",

    "mathieu": "ecg_baseline_mathieu.csv"

}
```

```
475 # --- Step 2: Load and process HRV feature matrices ---
476
477 driver_models = {}
478
479 driver_names = list(files.keys())
480
481 driver_data = []
482
483 for name in driver_names:
484
485     filepath = os.path.join(path, files[name])
486
487     X_driver = process_ecg_to_hrv_features(filepath)
488
489     print(f"{name} shape: {X_driver.shape}")
490
491     driver_data.append(X_driver)
492
493
494
495 # --- Step 3: Stack all data into one matrix ---
496
497 X = np.vstack(driver_data)
498
499 labels = np.concatenate([[i+1]*len(driver_data[i]) for i in
    ↪  range(len(driver_data))])
500
501
502
503 # --- Step 4: Standardize (Step 5 of Algorithm 1) ---
504
505 scaler = StandardScaler()
506
507 X_scaled = scaler.fit_transform(X)
508
509
510
511 # --- Step 5: PCA (Step 6 of Algorithm 1) ---
512
513 pca = PCA()
514
515 pca.fit(X_scaled)
516
517
518
519 explained_variance = pca.explained_variance_ratio_
520
521 cumulative_variance = np.cumsum(explained_variance)
522
523
524
525 threshold = 0.95
526
527 R = np.argmax(cumulative_variance >= threshold) + 1
528
529 print(f"Chosen R: {R} (explains {cumulative_variance[R-1]:.4f} variance)")
530
531
532
533 # Optional: Plot explained variance
```

```
534
535 plt.figure(figsize=(8, 4))
536
537 plt.plot(np.arange(1, len(explained_variance)+1), cumulative_variance,
    ↪  marker='o')
538
539 plt.axhline(y=0.95, color='r', linestyle='--', label='95% variance')
540
541 plt.axvline(x=R, color='g', linestyle='--', label=f'R = {R}')
542
543 plt.xlabel('Number of Components')
544
545 plt.ylabel('Cumulative Explained Variance')
546
547 plt.title('Explained Variance vs. Components')
548
549 plt.legend()
550
551 plt.grid(True)
552
553 plt.tight_layout()
554
555 plt.show()
556
557
558
559 # Fit PCA with R components
560
561 pca_R = PCA(n_components=R)
562
563 X_pca = pca_R.fit_transform(X_scaled)
564
565
566
567 # Store singular values and loading matrix
568
569 sigma_R = pca_R.singular_values_
570
571 V_R = pca_R.components_
572
573
574
575 print("Singular values (Σ_R):", sigma_R)
576
577 print("Loading matrix (V_R) shape:", V_R.shape)
578
579 print("Transformed data (X_pca) shape:", X_pca.shape)
580
581
582
583 def remove_outliers(values):
584
585     q1 = np.percentile(values, 25)
586
587     q3 = np.percentile(values, 75)
588
589     iqr = q3 - q1
590
591     lower_bound = q1 - 1.5 * iqr
592
```

```
    upper_bound = q3 + 1.5 * iqr

    return values[(values >= lower_bound) & (values <= upper_bound)]


def compute_control_limitstest(X_unscaled, scaler, pca_model, alpha=0.95):

    X_scaled = scaler.transform(X_unscaled)

    X_pca = pca_model.transform(X_scaled)

    X_reconstructed = pca_model.inverse_transform(X_pca)


    # T² statistic

    scores = X_pca

    eigenvalues = pca_model.explained_variance_

    T2_statistic = np.sum((scores**2) / eigenvalues, axis=1)


    # Q statistic

    residuals = X_scaled - X_reconstructed

    Q_statistic = np.sum(residuals**2, axis=1)


    # Remove outliers before thresholding

    T2_filtered = remove_outliers(T2_statistic)

    Q_filtered = remove_outliers(Q_statistic)


    # Calculate thresholds from filtered data

    T2_threshold = np.percentile(T2_filtered, alpha * 100)

    Q_threshold = np.percentile(Q_filtered, alpha * 100)


    return T2_threshold, Q_threshold, T2_statistic, Q_statistic



# --- Step 6: Compute control limits per driver ---

def compute_control_limits(X_unscaled, scaler, pca_model, alpha=0.95):

```

```
    # Reuse global scaler and PCA

    X_scaled = scaler.transform(X_unscaled)

    X_pca = pca_model.transform(X_scaled)

    X_reconstructed = pca_model.inverse_transform(X_pca)


    # T² statistic

    scores = X_pca

    eigenvalues = pca_model.explained_variance_

    T2_statistic = np.sum((scores**2) / eigenvalues, axis=1)


    # Q statistic (reconstruction error)

    residuals = X_scaled - X_reconstructed

    Q_statistic = np.sum(residuals**2, axis=1)


    # Control limits (thresholds)

    T2_threshold = np.percentile(T2_statistic, alpha * 100)

    Q_threshold = np.percentile(Q_statistic, alpha * 100)


    return T2_threshold, Q_threshold, T2_statistic, Q_statistic


# Apply to each driver

control_limits = []

for i, X_driver in enumerate(driver_data):

    T2_thresh, Q_thresh, T2_vals, Q_vals = compute_control_limits(X_driver,
     ↪  scaler, pca_R)

    control_limits.append({

        "driver_id": i,

        "T2_threshold": T2_thresh,

        "Q_threshold": Q_thresh,

        "T2_values": T2_vals,

        "Q_values": Q_vals
```

```
    })


# Save for algorithm 2

model_bundle = {

    "scaler": scaler,

    "pca": pca_R,

    "control_limits": control_limits

}


joblib.dump(model_bundle, os.path.join(model_dir, "driver_model_bundle.pkl"))


# --- Step 7: Print thresholds ---

for c in control_limits:

    print(f"\nDriver {c['driver_id'] + 1} Control Limits:")

    print(f"  T² threshold: {c['T2_threshold']:.3f}")

    print(f"  Q  threshold: {c['Q_threshold']:.3f}")

    print(f"  Mean T²: {np.mean(c['T2_values']):.3f}, Std T²:
     ↪  {np.std(c['T2_values']):.3f}")

    print(f"  Mean Q : {np.mean(c['Q_values']):.3f}, Std Q :
     ↪  {np.std(c['Q_values']):.3f}")


# --- Step 8: Plot histograms of T² and Q for each driver ---

for c in control_limits:

    driver_id = c["driver_id"] + 1

    plt.figure(figsize=(10, 4))



    plt.subplot(1, 2, 1)

    sns.histplot(c["T2_values"], kde=True, bins=20)

    plt.axvline(c["T2_threshold"], color='r', linestyle='--', label='T²
     ↪  threshold')

    plt.title(f'Driver {driver_id} T² Statistic')
```

```
    plt.xlabel("T²")

    plt.legend()



    plt.subplot(1, 2, 2)

    sns.histplot(c["Q_values"], kde=True, bins=20)

    plt.axvline(c["Q_threshold"], color='r', linestyle='--', label='Q threshold')

    plt.title(f'Driver {driver_id} Q Statistic')

    plt.xlabel("Q")

    plt.legend()



    plt.tight_layout()

    plt.show()


```

B.1.4. Python Code for Algorithm 2

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from scipy.signal import iirnotch, filtfilt, butter

import os

from ecgdetectors import Detectors

import math

from hrv import HRV

import neurokit2 as nk

from sklearn.preprocessing import StandardScaler

from sklearn.decomposition import PCA

import scipy.stats as stats

import seaborn as sns

from hrvanalysis import remove_outliers, remove_ectopic_beats,
 ↪ interpolate_nan_values

from hrvanalysis import get_time_domain_features, get_frequency_domain_features
```

```
30
31  import joblib
32
33
34
35
36
37  def process_ecg_to_hrv_features(filepath):
38
39
40
41      # === Load and Clean ===
42
43      df = pd.read_csv(filepath).iloc[1:]
44
45      esp_df = df.dropna(subset=[”esp_value”]).copy()
46
47
48
49      fs = 1 / np.mean(np.diff(esp_df['timestamp'].values))
50
51      print(f”Sampling rate (fs) is {fs:.2f} Hz”)
52
53
54
55      # === Filters ===
56
57      def notch_filter(data, notch_freq=50.0, fs=fs, Q=30):
58
59          b, a = iirnotch(notch_freq, Q, fs)
60
61          return filtfilt(b, a, data)
62
63
64
65      def bandpass_filter(data, lowcut=0.5, highcut=20, fs=fs, order=2):
66
67          nyq = 0.5 * fs
68
69          low = lowcut / nyq
70
71          high = highcut / nyq
72
73          b, a = butter(order, [low, high], btype='band')
74
75          return filtfilt(b, a, data)
76
77
78
79      def smooth(data, window_size=5):
80
81          return np.convolve(data, np.ones(window_size)/window_size, mode='same')
82
83
84
85      # === Split into windows ===
86
87      def split_by_time(df, window_sec):
88
89          windows = []
```

```
90
91          start_time = df['timestamp'].iloc[0]
92
93          end_time = math.ceil(df['timestamp'].iloc[-1])
94
95          current_start = start_time
96
97          while current_start + window_sec <= end_time:
98
99              current_end = current_start + window_sec
100
101             window = df[(df['timestamp'] >= current_start) & (df['timestamp'] <
                ↪  current_end)].reset_index(drop=True)
102
103             if not window.empty:
104
105                 windows.append(window)
106
107             current_start = current_end
108
109         return windows
110
111
112
113     def split_by_time_test(df, window_sec, step_sec):
114
115         windows = []
116
117         start_time = df['timestamp'].iloc[0]
118
119         end_time = math.ceil(df['timestamp'].iloc[-1])
120
121         current_start = start_time
122
123
124
125         while current_start + window_sec <= end_time:
126
127             current_end = current_start + window_sec
128
129             window = df[(df['timestamp'] >= current_start) & (df['timestamp'] <
                ↪  current_end)].reset_index(drop=True)
130
131             if not window.empty:
132
133                 windows.append(window)
134
135             current_start += step_sec  # <-- This creates the sliding behavior
136
137
138
139         return windows
140
141
142
143
144
145
146
147     # === R-peak Detection ===
```

```
148
149     def ecg_peaks(x, sampling_rate):
150
151         detectors = Detectors(sampling_rate)
152
153         search_interval = 150  # in samples
154
155
156
157         # Step 1: Pan-Tompkins Detection
158
159         R = np.asarray(detectors.pan_tompkins_detector(x))
160
161
162
163         # Step 2: Refine to local max
164
165         for j in range(R.size):
166
167             left = max(R[j] - search_interval, 0)
168
169             right = min(R[j] + search_interval, len(x))
170
171             local_peak = np.argmax(x[left:right])
172
173             R[j] = local_peak + left
174
175
176
177         # Step 3: Remove duplicates
178
179         R = np.unique(R)
180
181
182
183         # Step 4: Remove low peaks
184
185         peak_heights = x[R]
186
187         threshold = np.median(peak_heights) * 0.5
188
189         R = R[peak_heights > threshold]
190
191
192
193         # Step 5: Remove peaks too close together (< 300ms)
194
195         min_interval = int(0.3 * sampling_rate)
196
197         final_peaks = [R[0]]
198
199         for i in range(1, len(R)):
200
201             if R[i] - final_peaks[-1] >= min_interval:
202
203                 final_peaks.append(R[i])
204
205         return np.array(final_peaks)
206
207
```

```
208
209     # === Process Signal ===
210
211     esp_df['esp_scaled'] = (esp_df['esp_value'].astype(float) * 5) / 4096
212
213     esp_df['ecg_filtered'] =
         ↪  bandpass_filter(esp_df['esp_scaled'].values.astype(float), fs=fs)
214
215     signal = np.array(esp_df['ecg_filtered'], dtype=np.float32)
216
217     esp_df['smoothed'] = smooth(signal, window_size=5)
218
219
220
221     # windows = split_by_time(esp_df, 30)
222
223     windows = split_by_time_test(esp_df, 180, 10)
224
225     # print(f"Number of 30s windows: {len(windows)}")
226
227     print(f"Number of overlapping windows: {len(windows)}")
228
229
230
231     # === Detect R-peaks ===
232
233     r_peaks = ecg_peaks(esp_df['smoothed'].values, sampling_rate=fs)
234
235     r_peak_times = esp_df['timestamp'].values[r_peaks]
236
237     r_peak_amplitudes = esp_df['smoothed'].values[r_peaks]
238
239
240
241     # === Segment Peaks by Time ===
242
243     window_size = 180
244
245     start_time = esp_df['timestamp'].iloc[0]
246
247     end_time = esp_df['timestamp'].iloc[-1]
248
249
250
251     r_peaks_list, r_peak_times_list, r_peak_amplitudes_list = [], [], []
252
253     current_start = start_time
254
255     step_size = 10
256
257
258
259     while current_start < end_time:
260
261         current_end = min(current_start + window_size, end_time)
262
263         mask = (r_peak_times >= current_start) & (r_peak_times < current_end)
264
265
266
```

```
        window_r_peak_times = r_peak_times[mask]

        window_r_peak_amplitudes = r_peak_amplitudes[mask]

        window_r_peak_indices = r_peaks[mask]


        if len(window_r_peak_times) > 1:

            duration = window_r_peak_times[-1] - window_r_peak_times[0]

            if duration >= 120:

                r_peaks_list.append(window_r_peak_indices)

                r_peak_times_list.append(window_r_peak_times)

                r_peak_amplitudes_list.append(window_r_peak_amplitudes)

        current_start += step_size







    # === RR Interval Processing ===

    rr_intervals_list_ms = [np.diff(times) * 1000 for times in r_peak_times_list]


    nn_intervals_list_ms = []

    for rr in rr_intervals_list_ms:

        rr_no_outliers = remove_outliers(rr_intervals=rr, low_rri=300,
         ↪  high_rri=2000)

        rr_interp = interpolate_nan_values(rr_intervals=rr_no_outliers)

        nn = remove_ectopic_beats(rr_intervals=rr_interp, method="malik")

        nn_interpolated = interpolate_nan_values(rr_intervals=nn)

        nn_intervals_list_ms.append(nn_interpolated)


    # === HRV Feature Extraction ===

    mean_nni_list, rmssd_list, sdnn_list = [], [], []

    nni_50_list, mean_hr_list = [], []

    LF_list, HF_list, LF_HF_ratio_list, total_power_list = [], [], [], []


```

```
326
327     for nn in nn_intervals_list_ms:
328
329         if len(nn) > 2:
330
331             t_features = get_time_domain_features(nn)
332
333             f_features = get_frequency_domain_features(nn)
334
335
336
337             mean_nni_list.append(t_features['mean_nni'])
338
339             rmssd_list.append(t_features['rmssd'])
340
341             sdnn_list.append(t_features['sdnn'])
342
343             nni_50_list.append(t_features['nni_50'])
344
345             mean_hr_list.append(t_features['mean_hr'])
346
347
348
349             LF_list.append(f_features['lf'])
350
351             HF_list.append(f_features['hf'])
352
353             LF_HF_ratio_list.append(f_features['lf_hf_ratio'])
354
355             total_power_list.append(f_features['total_power'])
356
357         else:
358
359             mean_nni_list.append(np.nan)
360
361             rmssd_list.append(np.nan)
362
363             sdnn_list.append(np.nan)
364
365             nni_50_list.append(np.nan)
366
367             mean_hr_list.append(np.nan)
368
369             LF_list.append(np.nan)
370
371             HF_list.append(np.nan)
372
373             LF_HF_ratio_list.append(np.nan)
374
375             total_power_list.append(np.nan)
376
377
378
379     # === Plot ECG Signal with R-peaks ===
380
381     plt.figure(figsize=(12, 5))
382
383     plt.plot(windows[0]['timestamp'], windows[0]['smoothed'], color='blue',
      ↪  label='Smooth ECG')
384
```

```
385     plt.plot(r_peak_times_list[0], r_peak_amplitudes_list[0], 'ro',
        ↪  label='R-peaks')
386
387     plt.title("ECG Signal with R-Peaks")
388
389     plt.xlabel("Time (s)")
390
391     plt.ylabel("Voltage (V)")
392
393     plt.legend()
394
395     plt.grid(True)
396
397     plt.tight_layout()
398
399     plt.show()
400
401
402
403     # === Plot HRV Over Time ===
404
405     window_times = np.array([start_time + i * step_size + window_size / 2 for i
        ↪  in range(len(mean_hr_list))])
406
407
408
409     plt.figure(figsize=(10, 5))
410
411     plt.plot(window_times, mean_hr_list, marker='o', label='Mean HR')
412
413     plt.xlabel('Time (s)')
414
415     plt.ylabel('Heart Rate (bpm)')
416
417     plt.title('Mean Heart Rate Over Time')
418
419     plt.legend()
420
421     plt.grid(True)
422
423     plt.tight_layout()
424
425     plt.show()
426
427
428
429
430
431     # Algorithm 1:
432
433     # make matrix X with the features for the person
434
435     X_i = np.column_stack((mean_nni_list, sdnn_list, rmssd_list,
        ↪  total_power_list, nni_50_list, LF_list, HF_list, LF_HF_ratio_list))
436
437     print(f"X_i is {X_i}")
438
439
440
441     return X_i, mean_hr_list
```

```
442
443
444
445 path_1 =
     ↪ r"C:\Users\harim\Desktop\grip_code\BAP25\Measurement_code\sleep_recordings"
446
447 filename_1 = "sleep_peter.csv"
448
449 filepath_1 = os.path.join(path_1, filename_1)
450
451 X_matrix_sanne_unscaled, mean_hr_list = process_ecg_to_hrv_features(filepath_1)
452
453
454
455 model_path =
     ↪ r"C:\Users\harim\Desktop\grip_code\BAP25\Measurement_code\models_test\driver_model_bundle.
     ↪ pkl"
456
457 model_bundle = joblib.load(model_path)
458
459
460
461 scaler = model_bundle["scaler"]
462
463 pca_R = model_bundle["pca"]
464
465 control_limits = model_bundle["control_limits"]
466
467
468
469 print(control_limits)
470
471
472
473 # Step 5
474
475 # Preporcess X, mean = 0 and std = 1
476
477 X_scaled = scaler.transform(X_matrix_sanne_unscaled)
478
479
480
481 # Step 6: Compute T²[t] and Q[t]
482
483 X_pca = pca_R.transform(X_scaled)
484
485 X_reconstructed = pca_R.inverse_transform(X_pca)
486
487
488
489 # T²[t] using PCA eigenvalues
490
491 scores = X_pca
492
493 eigenvalues = pca_R.explained_variance_
494
495 T2_stats = np.sum((scores ** 2) / eigenvalues, axis=1)
496
497
498
```

```
# Q[t] is the squared reconstruction error

residuals = X_scaled - X_reconstructed

Q_stats = np.sum(residuals ** 2, axis=1)


print(f"T2_stats are {T2_stats}")

print(f"Q_stats are {Q_stats}")


# From sleep data


# From baseline control limits

driver_id = 5

T2_thresh = control_limits[driver_id]["T2_threshold"]

Q_thresh  = control_limits[driver_id]["Q_threshold"]

print(f"T2 threshold is {T2_thresh}")

print(f"Q threshold is {Q_thresh}")


# Initial state

state = "A"

C = []

tau = 0

tau_steps_required = 3  # 2 × 10s = 20s total


for t in range(len(T2_stats)):

    t2, q = T2_stats[t], Q_stats[t]

    t2_ok = t2 <= T2_thresh

    q_ok = q <= Q_thresh


    if (state == "A" and (not t2_ok or not q_ok)) or (state == "D" and (t2_ok and
     ↪  q_ok)):

        tau += 1

    else:
```

```
            tau = 0


        if tau >= tau_steps_required:
            state = "D" if state == "A" else "A"
            tau = 0


        C.append(state)


print(f"C is {C}")
print(len(mean_hr_list), len(C))


T2_pass_indices = np.where(T2_stats <= T2_thresh)[0]
print(f"T²-passing segments: {len(T2_pass_indices)} out of {len(T2_stats)}")


Q_pass_indices = np.where(Q_stats <= Q_thresh)[0]
print(f"Q-passing segments: {len(Q_pass_indices)} out of {len(Q_stats)}")


#  Example: Convert states to numeric (1 = Drowsy, 0 = Awake)
state_numeric = [1 if s == "D" else 0 for s in C]
time = np.arange(len(C)) * 10  # time in seconds


fig, ax1 = plt.subplots(figsize=(12, 5))


# Plot HR on primary axis
color = 'tab:blue'
ax1.set_xlabel('Time (s)')
ax1.set_ylabel('Heart Rate (bpm)', color=color)
ax1.plot(time, mean_hr_list, color=color, label="Mean HR")
ax1.tick_params(axis='y', labelcolor=color)
```

```
618
619 # Plot drowsiness state on secondary axis
620
621 ax2 = ax1.twinx()
622
623 color = 'tab:red'
624
625 ax2.set_ylabel('Drowsiness State', color=color)
626
627 ax2.plot(time, state_numeric, color=color, linestyle='--', label="Drowsy = 1,
    ↪ Awake = 0")
628
629 ax2.tick_params(axis='y', labelcolor=color)
630
631 ax2.set_yticks([0, 1])
632
633
634
635 # Legends
636
637 fig.tight_layout()
638
639 plt.title("Heart Rate and Drowsiness Detection Over Time")
640
641 plt.grid(True)
642
643 plt.show()
644
645
646
647
648
649 step_size = 10           # seconds between each sample
650
651
652
653 # Time axis
654
655 time_axis = np.arange(len(mean_hr_list)) * step_size
656
657
658
659 fig, ax1 = plt.subplots(figsize=(14, 6))
660
661
662
663 # Plot heart rate
664
665 ax1.plot(time_axis, mean_hr_list, label='Heart Rate', color='blue')
666
667 ax1.set_xlabel('Time (s)')
668
669 ax1.set_ylabel('Heart Rate (bpm)', color='blue')
670
671 ax1.tick_params(axis='y', labelcolor='blue')
672
673
674
675 # Shade regions where C == 'D'
676
```

```
677 in_drowsy = False
678
679 for i in range(len(C)):
680
681     if C[i] == 'D' and not in_drowsy:
682
683         start = time_axis[i]
684
685         in_drowsy = True
686
687     elif C[i] == 'A' and in_drowsy:
688
689         end = time_axis[i]
690
691         ax1.axvspan(start, end, color='red', alpha=0.15)
692
693         in_drowsy = False
694
695
696
697 # If still in drowsy at the end
698
699 if in_drowsy:
700
701     ax1.axvspan(start, time_axis[-1], color='red', alpha=0.15)
702
703
704
705 plt.title("Heart Rate and Drowsiness Detection")
706
707 plt.grid(True)
708
709 plt.tight_layout()
710
711 plt.show()
712
713
714
715
716
717
718
719 # Random test stuff
720
721 T2_drowsy_indices = np.where(T2_stats > T2_thresh)[0]
722
723 print(f"T²-drowsy segments: {len(T2_drowsy_indices)} out of {len(T2_stats)}")
724
725
726
727 Q_drowsy_indices = np.where(Q_stats > Q_thresh)[0]
728
729 print(f"Q-drowsy segments: {len(Q_drowsy_indices)} out of {len(Q_stats)}")
730
731
732
733 print("T²-drowsy segment indices:", T2_drowsy_indices)
734
735 print("Q-drowsy segment indices:", Q_drowsy_indices)
736
```

```
737
```