

Master in Human Centered Computing
„Internet of Things“

SS2016

Prof. Keller/Martinez/Steddin

- Masterprojekt -

PoSDBoS - Portable System to Detect Driver Drowsiness with Body Sensors.

vorgelegt von:

Paul Pasler

3. Semester

Kontaktadresse: paul.pasler@student.reutlingen-university.de

vorgelegt am: 31.08.2016

Zusammenfassung Fahrerassistenzsysteme sind aus modernen Fahrzeugen nicht mehr wegzudenken. Müdigkeitserkennung hilft Sekundenschlaf oder müdigkeitsbedingte Unachtsamkeit zu vermeiden und verhindert somit schwere Unfälle. Systeme mit Körpersensoren zeigen in verschiedenen Studien sehr genau Ergebnisse und erkennen Müdigkeit frühzeitiger als andere Ansätze. In der vorgelegten Arbeit wurde ein solches System mit einem Elektroenzephalogramm (EEG) umgesetzt und getestet. Hierfür wurden Testdaten aufgenommen, verarbeitet und mit einem künstlichen Neuronalen Netz klassifiziert, sodass der aktuelle Status des Fahrers „Wach“ oder „Müde“ unterschieden werden konnte.

Inhaltsverzeichnis

1 Einleitung	3
1.1 EEG Headset	5
2 Stand der Technik	7
3 Anforderungen	9
4 Rahmenbedingungen	12
4.1 Infrastruktur	12
4.2 Testdatenbeschaffung	13
5 PoSDBoS - Müdigkeitserkennung mit EEG-Headset	18
5.1 Datensammlung	18
5.2 Datenverarbeitung	21
5.3 Merkmalsextraktion	24
5.4 Klassifikator	26
6 Ergebnis	29
7 Fazit und Ausblick	31
8 Master Projekt	32
9 Literaturverzeichnis	34
10 Eidesstattliche Erklärung	38

Abbildungsverzeichnis

1.1	Skizze des Systemaufbaus	3
1.2	Problemstellung der Müdigkeitserkennung	4
1.3	Medizinisches EEG / EMOTIV Epoch	5
1.4	EEG Sensoranordnung	6
3.1	Schwerpunkte der Anwendung	10
4.1	Aufbau des Simulators	16
4.2	Versuchsaufbau Experiment	17
4.3	Driving Task Screenshot	17
4.4	Driving Task Spurwechsel	17
5.1	Aufbau der Anwendung	19
5.2	Einbettung der Anwendung	20
5.3	Sequenzdiagramm der Anwendung	21
5.4	Datenverarbeitungskette	22
5.5	Verarbeitungskette	23
5.6	Rohsignal und Histogramm	24
5.7	Fast Fourier Transformation	25
5.8	Schema eines Perceptrons / McCulloch-Pitts-Neurons	27
5.9	Schema eines Multi-Layer-Perceptrons	27

Tabellenverzeichnis

3.1	Anforderungen	11
5.1	KNN Ergebnis-Matrix	28

1 Einleitung

Fahrerassistenzsysteme sind innerhalb weniger Jahren von der Oberklasse in die Mittel- und Kleinwagenklasse vorgedrungen. Die Unternehmensberatung Strategy Analytics schätzt, dass in den nächsten Jahren sechs mal soviele Fahrerassistenzsysteme verbaut werden als heute [1]. Sie bieten dem Fahrer erhöhten Komfort (bspw. Tempomat) oder erhöhen die Sicherheit (bspw. Notbremsassistent). Laut der Boston Consulting Group, könnte der flächendeckende Einsatz von Fahrerassistenzsystemen die Unfallrate um ein knappes Drittel zurückgehen lassen [2].

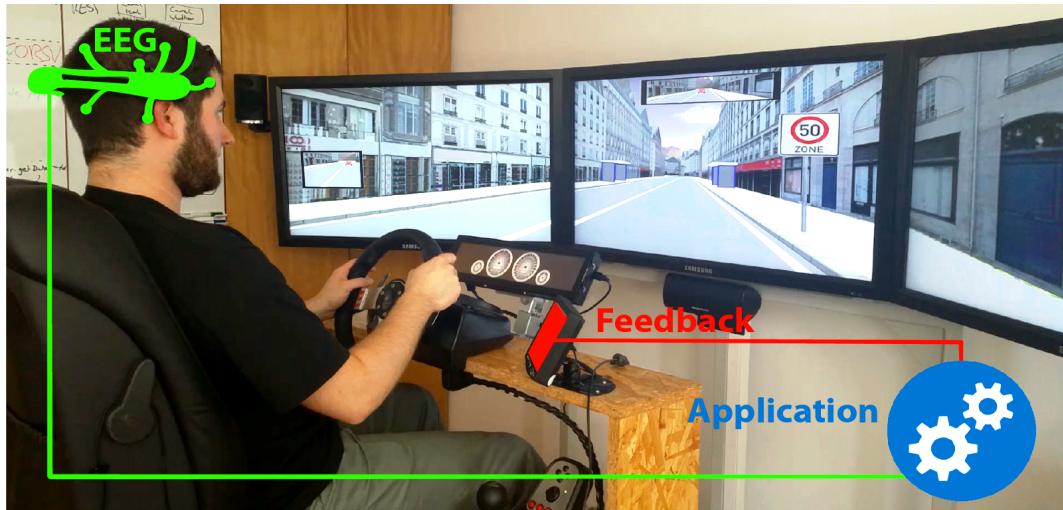


Abb. 1.1: Skizze des Systemaufbaus: Das EEG liefert Daten an die Applikation und ein Feedback-Device warnt den müden Fahrer. Bild zeigt den Fahrsimulator der Reutlingen University.

Zur Gruppe der Sicherheitsrelevanten Fahrerassistenzsysteme gehört auch die Müdigkeitserkennung. Beispielsweise rät die Müdigkeitserkennung „Attention Assist“ von Daimler dem Fahrer, zu gegebenem Anlass, eine Pause einzulegen und zeigt ein Kaffeesymbol im Cockpit an [3]. So wird müdigkeitsbedingte Unachtsamkeit oder Sekundenschlaf, die oftmals schwere Unfälle zur Folge haben, entgegengewirkt.

In Deutschland wurden 2015 rund 2,5 Mio. Unfälle polizeilich aufgenommen, die Zahl der Verkehrstoten liegt bei 3.450 [4]. Neben überhöhter Geschwindigkeit, zählt laut dem Deutschen Verkehrssicherheitsrat Müdigkeit zu den häufigsten Unfallursachen und ist damit für jeden fünften schweren Unfall verantwortlich [5]. Dies

verdeutlicht das Potential einer frühzeitigen Erkennung von Müdigkeit und einer Meldung an den Fahrer.

Für eine sichere und korrekte Erkennung von Übermüdung ergeben sich mehrere Problemstellungen (Abb. 1.2). Anzeichen von Müdigkeit müssen vom System genau analysiert werden, um eine sichere Aussage über die Fahrtauglichkeit des Fahrers treffen zu können. Erkennt das System eine gefährliche Situation, muss der Fahrer in geeigneter Weise darauf hingewiesen werden.

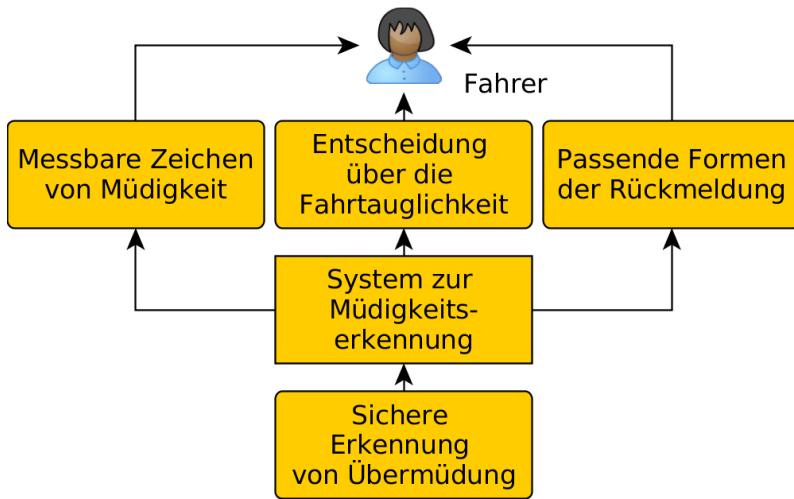


Abb. 1.2: Das System zur Müdigkeitserkennung gliedert sich in mehrere Teilproblemstellungen auf.

Daraus ergeben sich Anforderungen für ein multimodales System zur Müdigkeitserkennung (Abb. 1.1). Es existieren bereit diverse Systeme, denen es jedoch oftmals an Komfort und Portabilität mangelt. Ziel dieser Arbeit ist die Entwicklung einer solchen Anwendung mit einem EEG-Headset. Dessen Signale werden verarbeitet und mit einem KNN klassifiziert, um den Fahrer rechtzeitig vor Übermüdung und deren Folgen zu warnen. Statt einem klassischen EEG, wie es in der Medizin eingesetzt wird, wird ein EPOC Emotiv genutzt, dadurch soll die Beeinträchtigung des Fahrers verringert werden. Auch wenn sich selbst dieser Ansatz weniger für den Serienbetrieb eignet, wird das Handling deutlich verbessert. So kann das gesamte System beispielsweise für die Validierung von berührungslosen Systemen (bspw. Kamerabasiert) verwendet werden. Das ganze System soll leicht portierbar sein, um es in einem echten Fahrzeug testen zu können. Jedoch werden die notwendigen Testdaten im Rahmen des Projekts im Fahrsimulator der Reutlingen University aufgenommen.

Die Ausarbeitung gliedert sich folgendermaßen: im Kapitel 2 werden verschiedene Forschungsergebnisse zur Müdigkeitserkennung aufgezeigt und analysiert. Daraus

ergeben sich die Anforderungen an eine Anwendung zur Müdigkeitserkennung. Die Infrastruktur, die Beschaffung der Daten und die durchgeführten Versuche sind Thema von Kapitel 4.2. Die Implementierung eines portablen Systems zur Müdigkeitserkennung mit Körpersensoren wird im Kapitel 5 vorgestellt. Kapitel 6 beschreibt die Ergebnisse und leitet in Kapitel 7 zu den weiteren Schritten und dem Fazit über. Im folgenden Absatz werden Grundlagen für die kommenden Kapitel erläutert.

1.1 EEG Headset

Für das Projekt wurde ein EMOTIV Epoc+ EEG Headset¹ (Abb. 1.3, Quellen² ³) verwendet. Es besitzt 14 EEG Kanäle, sowie ein Gyroskop und sendet seine Daten via Bluetooth an den Rechner. Das Headset wird einfach über den Kopf gestülpt und positioniert. Am Kontakt zur Kopfhaut befindet sich einen Filz, der mit Kochsalzlösung befeuchtet wird.



Abb. 1.3: Links: Ein medizinisches EEG mit 64 Kanälen. Rechts: Das EMOTIV Epoc+ EEG Headset mit 14 Kanälen.

Die Sensoranordnung ist an das internationale 10-20 System [6] angelehnt. Hierbei handelt es sich um eine standardisierte relative Anordnung der Elektroden. Die Benennung der Sensoren sind aus Empfehlungen der „International Federation of Clinical Neurophysiology“ entnommen [7]. Abbildung (1.4) zeigt die Bezeichnung der 14 Sensoren.

¹<http://emotiv.com/epoc>

²https://en.wikipedia.org/wiki/Electroencephalography#/media/File:EEG_cap.jpg

³http://emotiv.com/wp-content/uploads/2016/04/epoc_hero_01_flipped.png

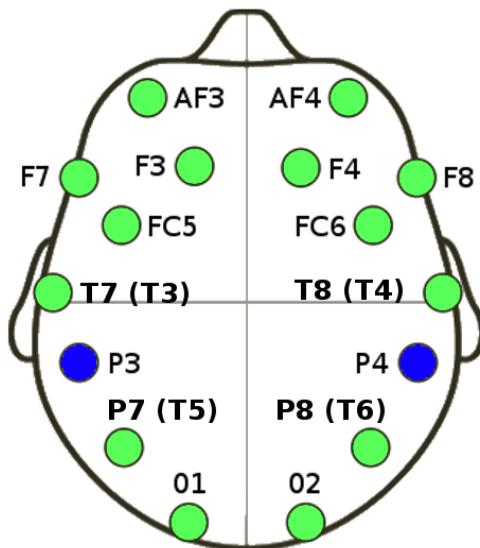


Abb. 1.4: Die 14 Kanäle (grün), sowie die beiden Qualitätssensoren (dunkelblau).

Die Rohdaten werden mit einer Abtastrate von 128Hz geliefert und enthalten den Signalwert, sowie die Qualität des Signals. Leider liefert die mitgelieferte SDK die Rohdaten nicht in Echtzeit, weshalb die Open-Source Lösung Emokit⁴ eingebunden wurde.

⁴<https://github.com/openyou/emokit>

2 Stand der Technik

Systeme zur Erkennung von Müdigkeit versuchen mit verschiedenen Parametern herauszufinden, ob sich die Person (der Fahrer) noch in einem aufmerksamen Zustand befindet. Die lässt sich in drei Bereiche unterteilen: Fahrverhalten, Computer-Vision (CV) und Körpersensoren. In der Praxis setzen Automobilhersteller wie Daimler [3] und Volkswagen, sowie Automobilzulieferer wie Bosch [8] auf die Analyse des Fahrverhaltens. Insbesondere das Abkommen von der Spur und ruckartiges Gegenlenken scheinen ein signifikantes Indiz für beginnende Übermüdung zu sein. Weiterhin sind externe Geräte und einige Apps für Smartphones erhältlich. Leider existieren kaum öffentlich zugängliche Arbeiten zu den verbauten Ansätzen von Müdigkeitserkennung, da es sich um interne Entwicklungen handelt. Wissenschaftliche Artikel sind ebenfalls selten [9].

Beim CV-Ansatz wird der Fahrer und die Straße mit Hilfe von Kameras beobachtet. Zhang et al. [10] stellen hierzu eine Applikation mit der Microsoft Kinect vor. Es wurden sowohl die Kopfpose, als auch der Augenstatus bestimmt. Bergasa et al. [11] extrahierten aus dem Bild einer Infrarot Kamera mehrere Features, wie beispielsweise den prozentualen Anteil von geschlossenen Augen (Percent eye closure, PERCLOS). Mit dieser Technik erreichten sie bei der Erkennung von Übermüdung eine nahezu hundertprozentige Erfolgsrate. Kamerabasierte Systeme schränken den Fahrer nicht ein, da kein direkter Kontakt zum Fahrer bestehen muss. Jedoch ist jede Kamera optischen Grenzen unterworfen (bspw. bei Nacht oder schlechtem Wetter).

Die beiden beschriebenen Bereiche werden in dieser Arbeit nur für die manuelle Markierung der aufgenommen Datensätzen genutzt. Der dritte Bereich für die Erkennung von Müdigkeit, arbeitet mit Körpersensoren bzw. deren Kombination (multimodal). Meist werden elektrische Spannung am und im Körper gemessen. Neben dem EEG, werden bspw. die Elektrokardiographie (EKG) oder Elektrookulografie (EOG) genutzt. Beim EKG wird die elektrische Aktivität des Herzmuskel erfasst, um bspw. die Herzfrequenz zu bestimmen. Das EOG misst die Bewegung der Augen, so lässt sich Blinzeln erkennen.

Ansätze mit einem EKG allein, zeigten in verschiedenen Arbeiten kein eindeutiges Ergebnis [12] [13]. Beide versuchten Informationen aus der Herzfrequenzvariabilität zu erhalten und diese zu klassifizieren. Khushaba et al. [14] versuchten EEG, EKG und EOG zu verbinden und verglichen verschiedene Kombinationen. Mit einem

„fuzzy wavlet“ basierten Algorithmus wurden die Signale aufbereitet und zeigten, dass ein EEG alleine bereits ausreicht. Die Kombination eines EEG mit EKG bzw. EOG verbesserte das Ergebnis nicht signifikant. Auch Johnson et. al [15] kamen zu der Erkenntnis, dass ein EEG ausreicht und das eingesetzte EOG nicht benötigt wird. Subasi [16] konnte mit einem EEG die Zustände „Wach“, „Schläfrig“ und „Schlafend“ unterscheiden. Die Wavelet-Transformation und das genutzte künstliche Neuronale Netz (KNN) führten zu einer Erkennungsrate von 93%. Vuckovic et al. [17] fanden den besten Algorithmus für die Initialisierung des KNNs: Der Learning Vector Quantization Algorithmus. Im Vergleich mit EEG-Experten erreichte das KNN eine Übereinstimmung von 90%. Huang et al. [18] nutzten ein Hidden Markov Modell zur Erkennung und erreichten eine gute Erfolgsrate. Lin et al. [19] nutzten die Unabhängige Komponenten Analyse (UKA) und Lineare Regression (LR) und konnten zeigen, dass hiermit bis zu 88% richtige Ergebnisse erzielt werden können.

Die betrachteten Arbeiten unterstreichen die Eignung von Körpersensoren um Müdigkeit zu erkennen. Die Stärken im Bereich Präzision und Korrektheit, im Vergleich zu Verhaltensanalyse oder CV-Techniken, zeigten sich in der Erkennungsrate. Den Komfort betreffend bleiben Körpersensoren jedoch hinter den anderen Ansätzen zurück.

Verhaltensanalyse oder CV-Technik werden für die Arbeit nicht berücksichtigt und treten nur bei der Analyse / Markierung der Testdaten in Erscheinung. Gähnen, häufiges Blinzeln oder Abkommen von der Fahrspur, können Hinweise auf Müdigkeit sein. Treten diese Anzeichen auf, können die entsprechenden EEG-Sequenzen gelabelt werden können. Aufgrund seiner höheren Genauigkeit, gegenüber EKG oder EOG, wird für die Anwendung ein EEG genutzt. Ein multimodales System ist vorbereitet, aber nicht umgesetzt. Die betrachteten Arbeiten zeigen in verschiedensten Ausführungen sehr gute Ergebnisse (~90% [19], [16]), sodass es sich beim EEG um eine aussichtsreiche Grundlage handelt. Für die Klassifizierung nutzen die meisten Ansätze ein Künstliches Neuronales Netz (KNN) [16] [17] [20] [21]. Weitere Ansätze nutzen die Lineare Diskriminanten Analyse [12] [14] oder ein SVM [22] [23]. Aufgrund der Häufigkeit in den anderen Arbeiten und der guten Bibliotheksunterstützung (PyBrain) wurde der Ansatz mit einem KNN weiter verfolgt. Die vorgestellten Lösungen arbeiten mit medizinischen oder selbst gebauten EEGs. Alle Ansätze wurden in einer simulierten Umgebung entwickelt und nie unter realen Bedingungen getestet. Aus der Analyse der verwandten Forschungsarbeiten ergeben sich Anforderungen an eine Anwendung zur Müdigkeitserkennung.

3 Anforderungen

Da es sich um ein sicherheitsrelevantes Fahrerassistenzsystem handelt, muss es in erster Linie präzise und korrekt funktionieren. Nicht ausgelöste Müdigkeitswarnungen wägen den Fahrer in Sicherheit, obwohl er evtl. nicht mehr in der Lage ist, sein Fahrzeug zu führen. Falsch ausgelöste Warnungen senken die Akzeptanz der Anwendung und führen im Extremfall zu deren Abschaltung. Zudem muss es robust gegen Störungen und Falscheingaben sein, es muss zu jeder Zeit gewährleistet sein, dass das System läuft bzw. den Fahrer im Fehlerfall rechtzeitig über den Status der Anwendung informieren.

Die Anwendung muss zudem nahezu in Echtzeit funktionieren und den Fahrer sofort über eine erkannte Müdigkeit informieren. Bei der Implementierung muss auf die Performance der Erkennung geachtet werden. Eine zu späte Meldung an den Fahrer könnte zu einem Unfall führen. Um das System möglichst flexibel zu machen, sollte es auf verschiedenen Plattformen lauffähig sein, auch hier muss eine verringerte Rechenleistung beachtet werden (bspw. auf einem Smartphone).

Um die Software möglichst flexibel bzw. unabhängig von der Hardware zu machen, sollten Datenquelle und Datenverarbeitung möglichst lose gekoppelt sein und sich leicht auf verschiedenen Systemen ausführen lassen. Das Hinzufügen von weiteren Quellen (bspw. EKG oder EOG) sollte vorbereitet werden.

Die Rückmeldung der Anwendung soll den Fahrer warnen, sodass er diese auf jeden Fall wahrnimmt, jedoch nicht in die Fahrsituation eingreift. Es sollte vom Fahrer mehr als Hinweis verstanden werden und nicht als Maßregelung, da dies die Akzeptanz wiederum mindern könnte. Die Anwendung soll ohne lange Einrichtung oder Interaktion des Fahrers funktionieren.

Der Komfort beim Fahren sollte möglichst hoch sein und die Sensoren sollten den Fahrer möglichst wenig beeinträchtigen. Mit einem medizinischen EEG mit 64 Pins und vielen Kabeln, ist das kaum möglich. Das Emotiv EEG lässt sich wie eine Mütze aufsetzen und überträgt seine Daten via Bluetooth - ein Komfortgewinn. Im Produktiveinsatz ist das dennoch zu unbequem und wird in dieser Form nicht in Serie gehen können. Weiterhin soll das System unter realen Bedingungen getestet werden können und sich daher leicht vom Fahrsimulator der Reutlingen University

in ein echtes Auto oder einen anderen Simulator portieren lassen. So können Störungen während einer richtigen Autofahrt (die sich nicht simulieren lassen) erkannt werden. In einem anderen Simulator kann die Anwendung zur Validierung und Verbesserung von anderen Systemen verwendet werden.

Für ein Projekt an einer Hochschule, das auch für Folgeprojekte genutzt werden soll, ergeben sich ebenso Anforderungen an die Codequalität, insbesondere an Lesbarkeit und Wartbarkeit. Die Funktionalität und der Aufbau der Anwendung muss sauber dokumentiert werden, sodass der Einstieg für neue Entwickler möglichst reibungslos verläuft.

Diese Anforderungen sind in Tabelle 3.1 zusammengefasst. Wichtigste Punkte sind Korrektheit, Portabilität und Komfort des Systems (Abb. 3.1). Wobei vor allem Portabilität und Komfort in den betrachteten Arbeiten bisher kaum berücksichtigt wurden.

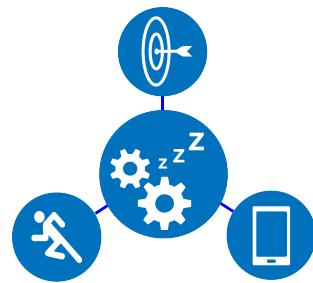


Abb. 3.1: Die Anwendung mit den Schwerpunkten: Korrektheit, Portabilität und Komfort

Tab. 3.1: Anforderungen

- 1 Die Anwendung muss Müdigkeit präzise und korrekt erkennen (> 90%)
- 2 Die Anwendung muss robust gegen Fehler / Fehleingaben sein
- 3 Die Anwendung muss den Fahrer über Störungen informieren
- 4 Die Anwendung muss den Fahrer so schnell wie möglich über eine erkannte Müdigkeit informieren (nahezu Echtzeit)
- 5 Die Anwendung muss sich in andere Umgebungen (Simulator oder echtes Fahrzeug) portieren lassen
- 6 Das eingesetzte EEG soll sich gut handhaben lassen und den Fahrer möglichst wenig beeinträchtigen
- 7 Die Anwendung soll auf möglichst vielen Plattformen lauffähig sein (verschiedene Betriebssysteme und Gerätetypen)
- 8 Die Anwendung soll möglichst wenig Ressourcen verbrauchen (siehe 5.)
- 9 Die Anwendung soll sich nahtlos in die Fahrsimulator-Umgebung einfügen
- 10 Datenakquise und Verarbeitung sollen auf getrennten Systemen durchgeführt werden können
- 11 Die Meldung über erkannte Müdigkeit soll vom Fahrer bemerkt werden, ihn aber nicht ablenken oder erschrecken
- 12 Die Anwendung (Module, Klassen, Methoden) soll komplett dokumentiert sein
- 13 Die Anwendung (Module, Klassen, Methoden) soll komplett durch Tests (mind. Unit-Tests) abgesichert sein

4 Rahmenbedingungen

Aus den Anforderungen des vorhergehenden Kapitels und der vorgegebenen Infrastruktur folgen die Rahmenbedingungen der Implementierung. Das Experiment zur Beschaffung der Testdaten wird dann im nächsten Abschnitt beschrieben.

4.1 Infrastruktur

Die Entwicklung der Anwendung (Experiment, Implementierung und Test) werden im Fahrimulator des IoT Labs¹ der Reutlingen University (Abb. 4.1) durchgeführt. Der Fahrimulator besteht aus einem Simulationsrechner mit drei 20"Monitoren, einem echten Fahrersitz, Lenkrad, Schaltung und Pedalen. Für die Simulation wird die Open-Source Software OpenDS² genutzt. Per TCP/IP werden alle notwendigen Fahrzeugdaten vom Simulationsrechner auf den Datensammler und dort in eine virtuelle Steuergerät-Software (Vector CANoe³) geschickt. Über einen CAN-Bus können die Daten dann wieder, mittels einer Schnittstelle (CAN-Interface), ausgelesen werden. Die eigentliche Applikation wird auf dem Embeddedrechner ausgeführt.

Die Anwendung selbst wird in Python 2.7⁴ realisiert. Python ist OpenSource, lässt sich leicht lernen und der Code ist, dank des ausgefeilten Sprachkonzepts, gut lesbar. Wie schon angemerkt, ist dies für Hochschulprojekte eine wichtige Eigenschaft, da es häufig zu Personalwechseln kommt. Für Python existieren eine Vielzahl an Bibliotheken für wissenschaftliche Anwendungen. Für das Projekt werden unter anderem die SciPy⁵, welche sich an der Funktionalität von Matlab orientiert, oder die MachineLearning-Bibliothek PyBrain⁶ genutzt. Python läuft auf allen gängigen Betriebssystemen (Windows, Mac oder Linux), wenn der passende Python-Interpreter installiert ist. Weiterhin existieren Scripte, um eine Anwendung auch auf anderen Plattformen lauffähig zu machen (Android, iPhone, etc.). Als Scriptsprache ist Python nicht ganz so schnell, wie eine kompilierte Sprache (Java, C#), liefert jedoch ausreichende Performance. Weiterhin kann eine Python

¹<http://iotlab.reutlingen-university.de/>

²<https://www.opensds.eu>

³https://vector.com/vi_canoe_de.html

⁴<https://www.python.org>

⁵<http://www.scipy.org>

⁶<http://pybrain.org>

Anwendung in ByteCode kompiliert werden, um die Geschwindigkeit weiter zu steigern.

Die komplette Anwendung ist mit docstrings (Dokumentation im Code) versehen, welche über eine HTML Seite abrufbar sind. Weiterhin sind alle wichtigen Codestellen durch Unitests abgesichert. Um den Einstieg in komplexe Themengebiete zu erleichtern, sind einfach Beispiele und Visualisierungen implementiert. Der komplette Code ist unter Versionskontrolle in einem git-Repository.

Die Anwendung fügt sich nahtlos in die Simulationsumgebung ein, kann aber auch Standalone betrieben werden. Das CAN-Interface kann die EEG-Rohdaten direkt via http empfangen und auf den CAN-Bus legen, als wäre das EEG ein Fahrzeugsensor. Die EEG-Daten können von der Anwendung wieder vom CAN-Bus gelesen werden und dann entsprechend weiterverarbeitet werden. Der http-Server kann auch direkt angesprochen werden. Somit sind Datenbeschaffung (EEG-Rohdaten) und Verarbeitung getrennt und können auf unterschiedlichen Geräten ausgeführt werden.

4.2 Testdatenbeschaffung

Um Daten von übermüdeten Fahrern zu erhalten, kann natürlich kein Versuch im Straßenverkehr durchgeführt werden. Die Fremd- und Eigengefährdung ist zu groß. Darum finden die Versuche in der Simulationsumgebung der Reutlingen University statt. Für das Experiment wird im Fahrsimulator eine Nacht-Autobahnfahrt simuliert. Verschiedene Studien [24], [25] legen nahe, dass sich Simulationen zwar von der Realität unterscheiden, dass die Ergebnisse jedoch trotzdem valide und brauchbar sind.

Für den Versuch werden neben den rohen EEG Signalen, auch die Fahrzeugdaten, sowie ein Video der Simulation und des Fahrers aufgenommen. Weiterhin kann der Versuchsleiter Besonderheiten protokollieren (Abb. 4.2).

Im Versuch sollte der Fahrer, mit fortschreitender Zeit, immer mehr eindeutige Anzeichen von Müdigkeit zeigen. Dies kann durch verschiedene Versuchspараметer begünstigt werden. So zeigt eine Studie, dass Unfälle meist zwischen 2:00 - 6:00, sowie 14:00 - 16:00 Uhr passieren [25]. Auch die Schlafmenge von weniger als 6 Stunden in der Nacht vor dem Experiment erhöht die Chance auf Anzeichen [24]. Das Geschlecht oder Alter der Probanden ist nicht relevant. Vor dem Experiment

sollten jedoch keine Drogen, Alkohol oder Kaffee eingenommen werden. Ein Führerschein ist von Vorteil, aber nicht zwingend notwendig.

Auch die Teststrecke (Abb. 4.3) trägt zur Erhöhung der Müdigkeit bei. Monotone Autobahnfahrten, die größtenteils geradeaus verlaufen, ohne andere Verkehrsteilnehmer und konstanter Geschwindigkeit führen eher zu einer Ermüdung. Nach diesen Kriterien wurde eine endlose zweispurige Autobahnkarte mit einer Geschwindigkeit von konstant 130Kmh erstellt. Sie findet zudem nachts im Dunkeln statt, was als besonders anstrengend für die Augen gilt. Die Simulation ist auf Bildschirm vermutlich noch anstrengender als eine echte Aussicht durch ein Autofenster.

Für einen Versuch werden 40 Minuten angesetzt. Fünf Minuten werden für eine kurze Einführung und Einrichten des EEGs benötigt, 30 Minuten für die Testfahrt und wieder fünf Minuten für eine kurze Selbsteinschätzung mit Fragebogen.

Anhand der aufgenommenen Daten werden nun Stellen gesucht, an denen die Testperson eindeutige Anzeichen von Müdigkeit zeigt. Eindeutig sind Verhaltensweisen wie häufiges Gähnen und Einnicken (Kopf fällt nach vorn) - diese Merkmale werden häufig in CV-Ansätzen genutzt. Auch Verhaltensmerkmale wie von der Spur abkommen und heftiges Gegenlenken oder deutliche Veränderungen der Geschwindigkeit können Anzeichen für eine Unachtsamkeit aufgrund von Müdigkeit sein. Diese Stellen werden dann in den EEG Daten mit dem Label „Müde“ markiert, alle anderen mit „Wach“. Die EEG Sequenzen können dann auf eindeutige Varianzen untersucht werden.

Das Experiment wurde mit 4 Personen (m / w) durchgeführt. Das Einrichten des EEG-Headsets (alle Sensoren auf hoher Qualität) war aufwändiger als erwartet, klappte jedoch bei allen Personen. Aus organisatorischen Gründen, fand nur ein Teil der Experimente nachmittags statt, der Rest fuhr morgens. Die Teilnehmer hatten im Vorfeld verschieden lang geschlafen und bewerteten auch die jeweilige Schlafqualität unterschiedlich. Die Testpersonen gaben an, vor dem Experiment eher wach als müde zu sein. Alle Fahrer gaben an, schon einmal übermüdet gefahren zu sein, waren sich aber nicht einig, ob sie sich eine Müdigkeitserkennung in einem Neuwagen bestellen würden.

Nach der Testfahrt gaben alle Teilnehmer an, müder als vorher gewesen zu sein. Dies zeigt sich auch an visuellen Merkmalen in den letzten Minuten der Testfahrt. Wobei die Ausprägungen von Müdigkeit bei den einzelnen Fahrern durchaus unterschiedlich war. Visuelle Anzeichen von Müdigkeit waren den Testpersonen bekannt

(Gähnen, häufige Positionswechsel, häufiges Blinzeln, trockene Augen) und konnten diese an sich selbst beobachten. Auch die Fahrweise zeigte teilweise deutliche Anzeichen von Unkonzentriertheit (Abkommen von der Spur, Schlangenlinien, zu niedrige / hohe Geschwindigkeit) Abb. 4.4.

Die Eigen- und Außenwahrnehmung bestätigt die Hypothese, dass es während des Experiments zu einem Abfall der Wachheit kommt und sich dies an Hand der erwarteten Merkmale zeigte. Die Veränderungen der Aufmerksamkeit muss nun im EEG Signal erkannt und extrahiert werden.

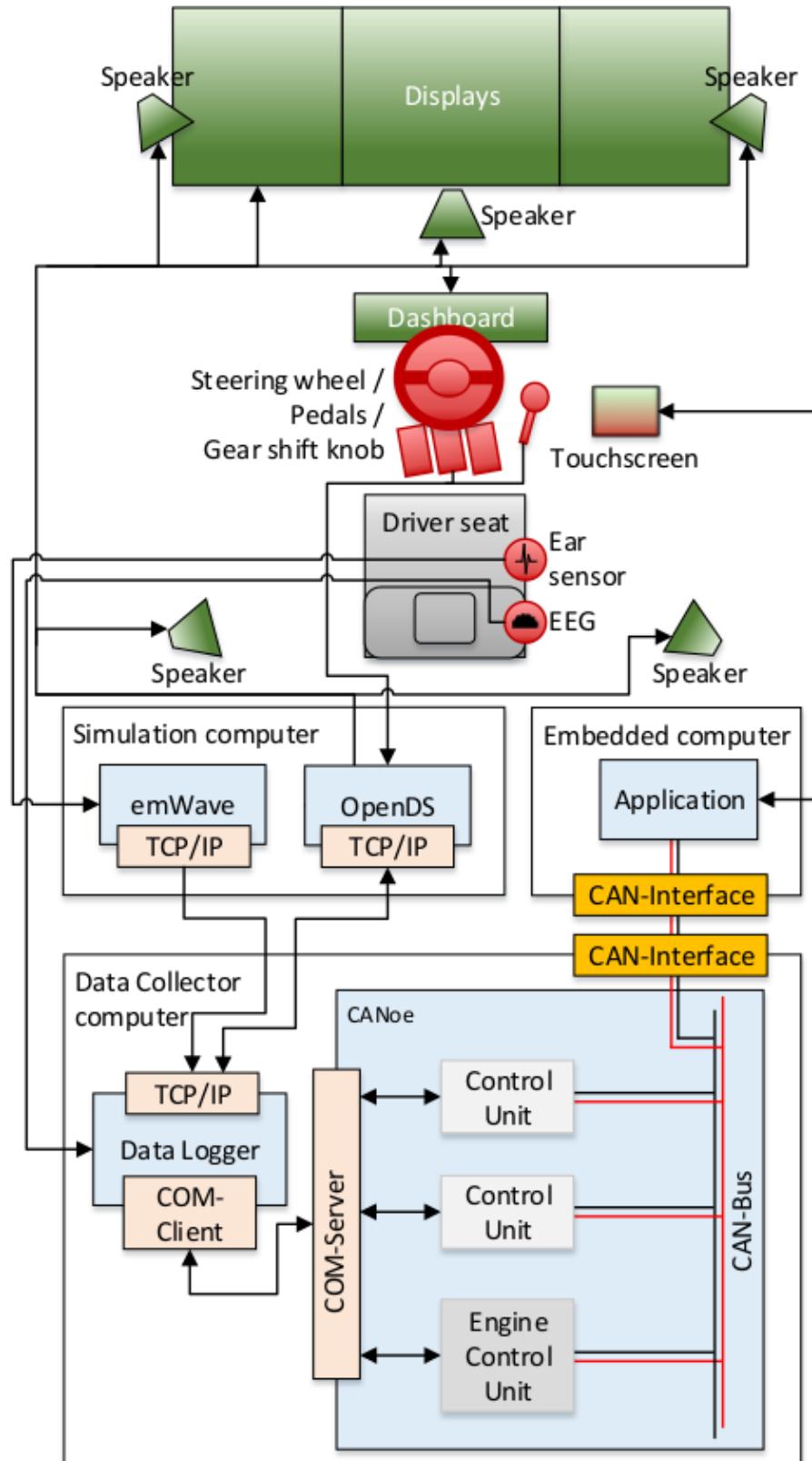


Abb. 4.1: Der Aufbau des Simulators der Reutlingen University mit den drei Rechnern für die Simulation, Datensammlung und Applikation.

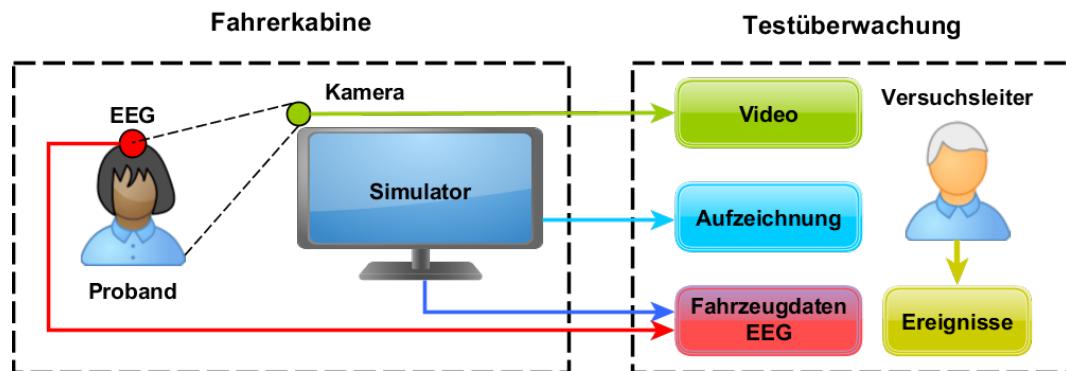


Abb. 4.2: Der Versuchsaufbau für die Erkennung von Müdigkeit mit den Datenströmen aus der Fahrerkabine zur Testüberwachung.



Abb. 4.3: Die Autobahnkarte für den Versuch verläuft endlos geradeaus und spielt nachts bei konstanter Geschwindigkeit.

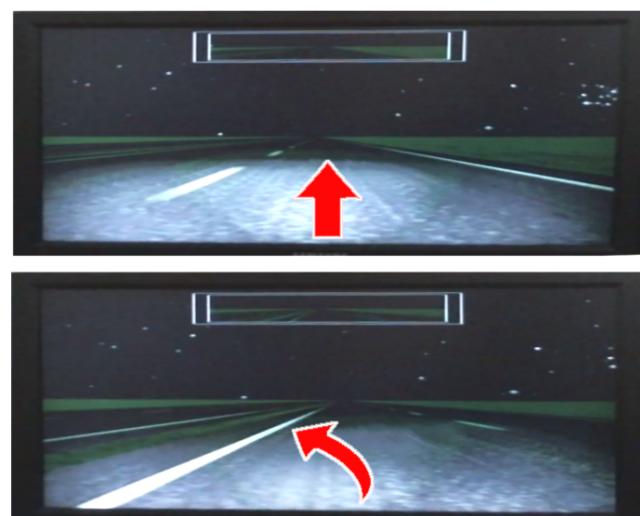


Abb. 4.4: Oben: Am Anfang fährt der Fahrer noch geradeaus auf der rechten Spur. Unten: gegen Ende kommt der Fahrer sogar von der linken Spur ab und fährt beinahe auf den Mittelstreifen

5 PoSDBoS - Müdigkeitserkennung mit EEG-Headset

Abbildung 5.1 zeigt die Gesamtübersicht des entwickelten Systems. Alle blauen Klassen der Anwendung wurden in Python 2.7, der Fahrsimulator in Java und seine Schnittstellen C# implementiert. Die einzelnen Datenströme sind grün gekennzeichnet, bedingte Abzweigungen, für alternative Wege, in gelb. Die Anwendung läuft verteilt auf dem DataCollector und dem Embedded PC. Die Datenquelle befindet sich auf dem DataCollector, von dort werden die Daten via CAN-Bus an den Embedded PC übertragen und verarbeitet (Abb. 5.2).

Die Daten werden über mehrere Threads übertragen, die oberen Klassen repräsentieren jeweils einen Thread. An unkritischen Stellen passiert dies per direktem Zugriff oder einem Callback (Abb. 5.3). Die Daten werden von der Emokit Klasse per Methodenaufruf geholt. Der DataCollector überträgt diese Daten an 1 .. n SignalWindows. Derzeit sind 2 SignalWindows implementiert, diese Zahl lässt sich erweitern. Wenn ein SignalWindow voll ist (128 Werte \equiv 1s) wird ein Callback ausgelöst, der direkt an den FeatureExtractor weitergegeben wird. Die Signal Sequenzen werden dort in eine threadsichere Queue eingefügt und 1 .. n Processing-Chains verarbeiten die Daten. Die Abarbeitung der ProcessingChain ist potentiell aufwändig und ist mit zwei Thread-sicheren Queues umgesetzt. Sollte die Verarbeitung zu lange dauern, könnten weitere ProcessingChains hinzugefügt werden. Der FeatureExtractor reicht diese Daten auf einer weiteren Queue zur Hauptanwendung, wo die Klassifizierung blockierend angestoßen wird und das Ergebnis zum Ausgabebildschirm geleitet wird. Der FeatureExtractor kann zu einem späteren Zeitpunkt eine Aggregation der Daten vornehmen oder sich um die Verwaltung der ProcessingChains kümmern.

Abschnitt 5.1 befasst sich mit den Rohdaten und deren Weiterreichung im System. Die Verarbeitung der Rohdaten ist Thema von Abschnitt 5.2. Im folgenden Abschnitt werden daraus die passenden Merkmale extrahiert. In Abschnitt 5.4 wird die Arbeitsweise des Klassifikators näher beleuchtet.

5.1 Datensammlung

Das EEG Headset schickt enkodierte Byte-Sequenzen via Bluetooth, an die proprietären Emotiv Premium Libraries oder die Open-Source Lösung Emotkit (vgl.

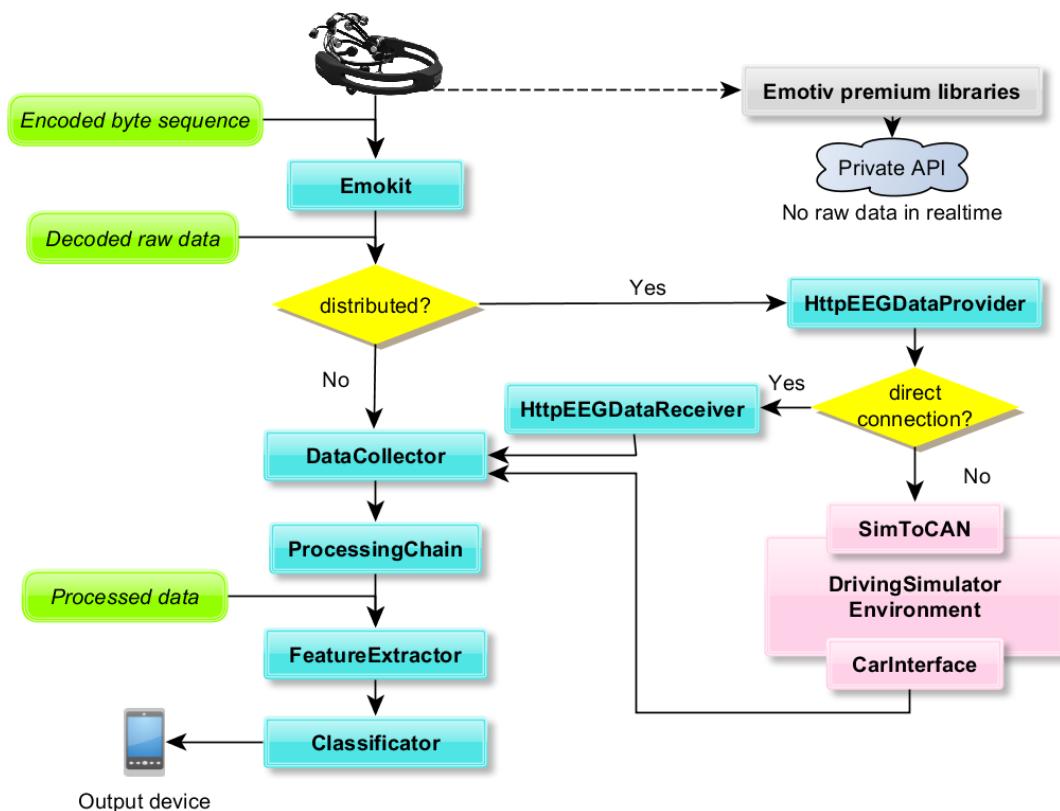


Abb. 5.1: Der Aufbau des entwickelten System zur Müdigkeitserkennung. Grün: Datenströme, blau: Python Klassen der Anwendung, gelb: bedingte Abzweigungen, rosa: Klassen der Fahrsimulatorumgebung

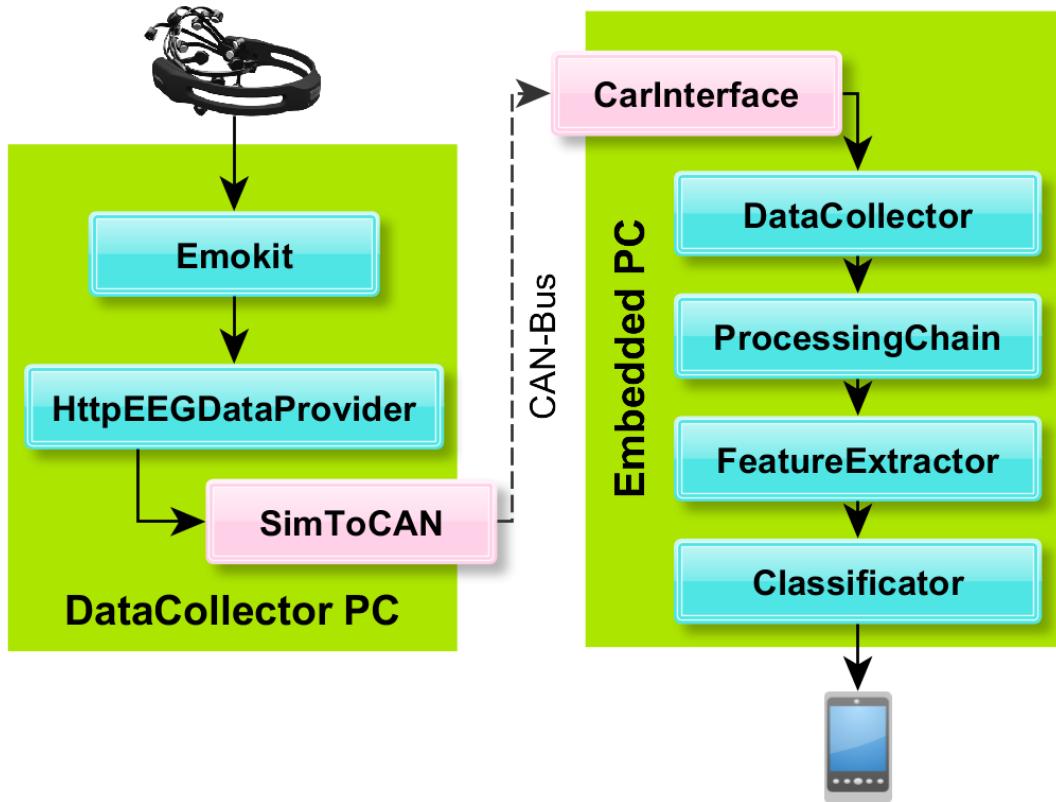


Abb. 5.2: Datenquelle und Verarbeitung sind verteilt im Fahrsimulator eingebettet. Die Übertragung erfolgt via CAN-Bus.

1.1). Die Emokit Klasse wurde für das Projekt leicht modifiziert, sodass sie sich ins System einfügt. So wurde Unterstützung für das neueste EPOC+ Modell implementiert, sowie die Möglichkeit, gespeicherte Testdaten aus dem TableReader zu versenden (Demo-Modus ohne Headset). Die dekodierten Rohdaten enthalten 14 EEG Kanäle mit Wert und Qualität, die Gyroskopwerte in X- und Y-Richtung und einen Zeitstempel (Abb. 5.4). Mit einer Abtastrate von 128Hz sind das $128 * 16 = 2048$ Werte pro Sekunde. Diese Daten können als CSV gespeichert werden, sowie an einen HTTP-Server oder direkt an den DataCollector übergeben werden. Über den Server integriert die SimToCAN Anwendung via HTTP die EEG Daten in den Fahrsimulator und das virtuelle Steuergerät.

Der DataCollector kann Daten direkt oder aus einem HTTP-Client empfangen. Die Schnittstelle zum CAR-Interface ist vorbereitet. So ist es möglich die Datensammlung und die Datenverarbeitung auf verschiedenen Rechnern durchzuführen.

Die Aufgabe der DataCollector-Klasse ist es, die einzelnen Signale in Sequenzfenstern von 128 Signalwerten zu aggregieren. Das ist die Abtastrate des EEG-Headsets und entspricht somit in etwa den Signalen einer Sekunde. Es sind zwei dieser Fenster

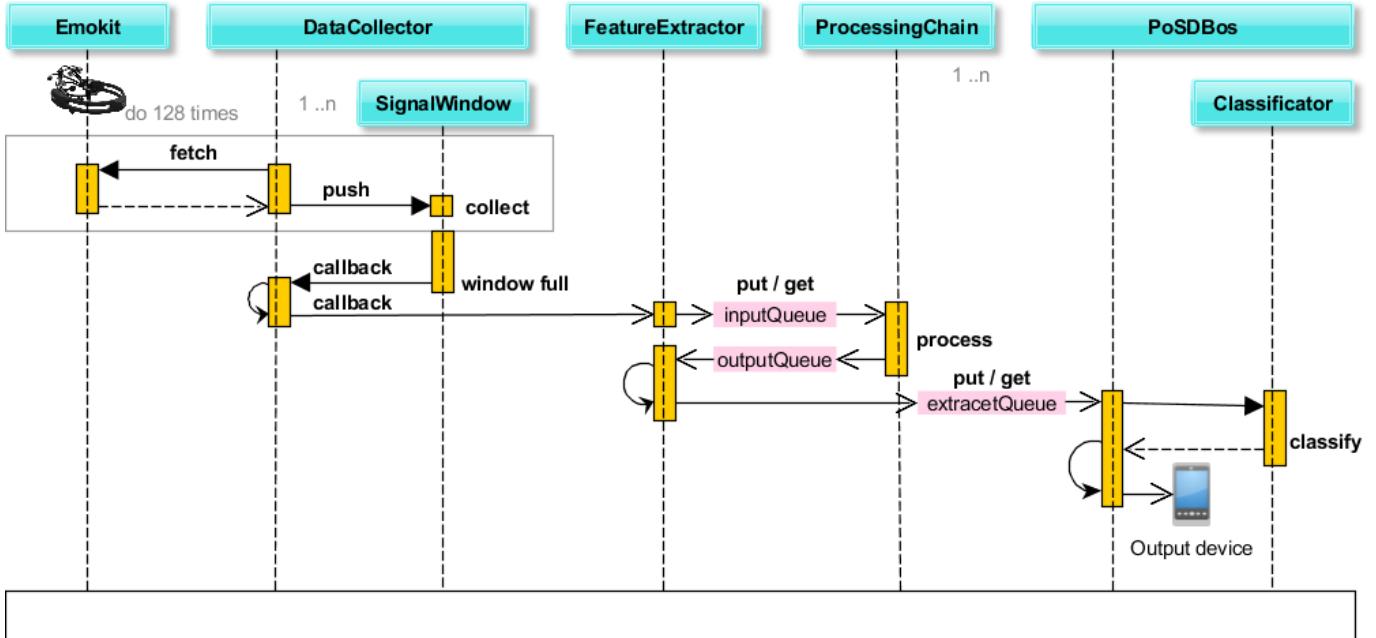


Abb. 5.3: Die Anwendung ist in mehrere Threads unterteilt. SignalWindow und Processing Chain können mehrere Instanzen haben.

implementiert und sie überschneiden sich in der Hälfte. So ist gewährleistet, dass signifikante Stellen nicht verloren gehen. Die Fensterfunktion ist ein simples Rechteck, sodass alle Werte gleich gewichtet werden. Eine andere Fensterfunktion bspw. mit Glockenkurvenverlauf (Hamming- oder Hann-Fenster) wäre einfach einzubauen. Auf den ersten Blick ist die Fensterlösung eine Verdopplung der Daten, jedoch fügt der DataCollector nur konfigurierte Kanäle hinzu, sodass sich die Datenmenge wieder deutlich reduzieren lässt. Im folgenden Abschnitt werden die gefilterten Sequenzfenster nun verarbeitet und aufbereitet.

5.2 Datenverarbeitung

Die Signalsequenzen durchlaufen eine Verarbeitungskette (Abb. 5.5) mit den Schritten Bandpassfilterung, Artefaktentfernung, Normalisierung, Fouriertransformation und Aufteilung nach Frequenzbändern.

Um Störungen zu eliminieren wurde das Signal zu Beginn außerhalb des Bereichs von 0.53Hz - 50Hz eliminiert. Dies war eine Empfehlung aus einer Antwort auf ResearchGate [26] und zeigt bei Tests gute Ergebnisse. Werte über 50Hz abschneiden macht Sinn, da diese Werte für die Müdigkeitserkennung keine Rollen spielen (β - und γ -Wellen). Signale unterhalb 0.53Hz ebenfalls zu entfernen hängt mit elektrischen Störungen zusammen. Der Bandpassfilter zentrierte das Signal zudem.

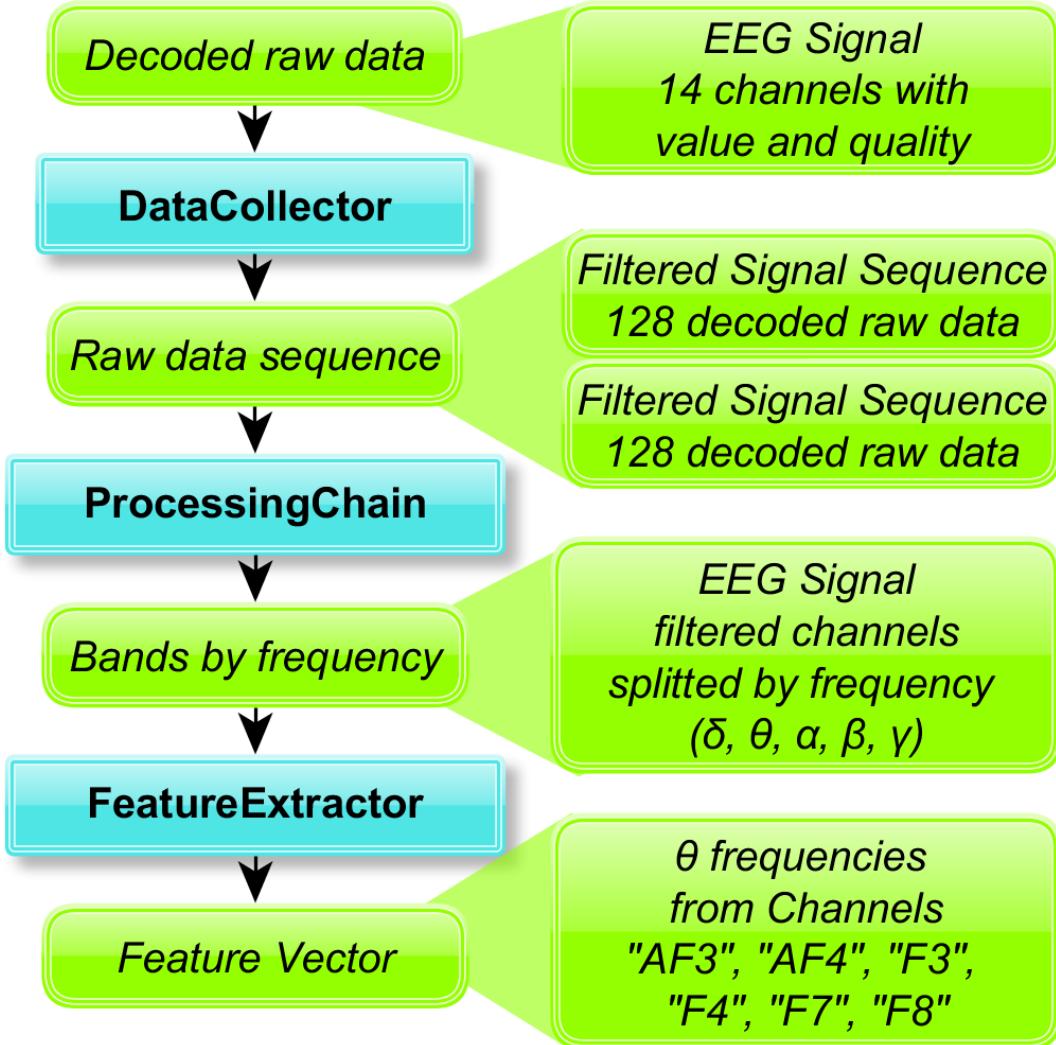


Abb. 5.4: Die Daten werden verarbeitet und dabei immer weiter reduziert.

Für die Anwendung wurde hierfür ein Butterworth-Filter [27] eingesetzt. Gleichung 5.1 zeigt die Übertragungsfunktion mit A_0 : Gleichspannungsverstärkung, $\Omega = \frac{f}{f_g}$: auf Grundfrequenz normierte Frequenz und n : Ordnung des Filters. Der Butterworth-Filter verläuft nahe Eins im gewünschten Bereich, fällt an den Grenzen ab und stellt sicher, dass das Signal an den Grenzen um $\frac{1}{\sqrt{2}} \approx 0.7071$ gemindert wird. Je höher die Ordnung, desto steiler geht die Funktion durch die angegebenen Grenzen. Als analoger Filter lässt er sich gut in Hardware realisieren.

$$|A|^2 = \frac{A_0^2}{1 + k_{2n}\Omega^{2n}} \quad (5.1)$$

Bei der Analyse der Histogramme der Signalwerte, zeigte sich eine Häufung der Amplituden im Bereich von -100 bis 100. Werte außerhalb dieses Bereichs wurden als Artefakte angesehen und herausgefiltert bzw. als ungültig markiert (Abb. 5.6).

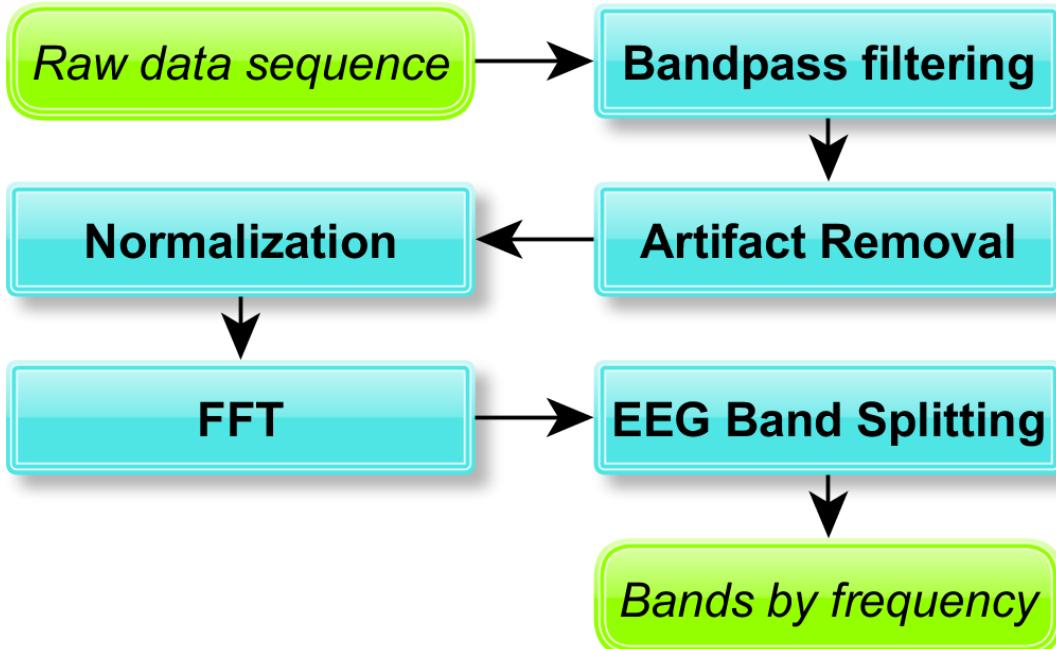


Abb. 5.5: Die Signalsequenzen durchlaufen eine Verarbeitungskette

Die gefilterten Signale werden dann auf einen Bereich von -1 bis 1 normalisiert, dazu werden sie durch den jeweiligen Maximalwert bzw. Betrag des Minimalwertes geteilt (in diesem Fall 100). Dieses Vorgehen stellt sicher, dass die absolute Amplitude keinen Einfluss auf die Gewichtung im Klassifikator hat. Zudem wird die Datenmenge wiederum reduziert.

Im letzten Schritt wird das Signal in seine Frequenzbänder (EEG-Bänder) unterteilt. Dazu wird es aus dem Zeitbereich in das Frequenzspektrum überführt und bestimmte Frequenzbereiche extrahiert. Diese EEG-Bänder gliedern sich in folgende Bereiche und werden nach griechischen Buchstaben benannt: δ : 0,5 - 4Hz, θ : 4 - 8Hz, α : 8 - 13Hz, β : 13 - 30Hz, γ : ab 30Hz. Die Frequenzbereiche sind nicht einheitlich definiert und variieren unter Umständen, sie sind aus Empfehlungen der IFCN entnommen [7]. Den Frequenzbändern werden verschiedene Eigenschaften zugesprochen [28–30]. Delta-Wellen treten bei Erwachsenen häufig in der traumlosen Tiefschlafphase auf. Theta-Wellen zeigen sich bei Schläfrigkeit und leichtem Schlaf. Mit leichter Entspannung und entspannter Wachheit (mit geschlossenen Augen) werden Alpha-Wellen assoziiert. Beta-Wellen treten während der REM-Schlafphase oder unter Einwirkung von Psychopharmaka auf. Gamma-Wellen gehen häufig mit starker Konzentration oder Meditation einher.

Um die einzelnen Frequenzbänder zu erhalten wird eine Fast-Fourier-Transformation [31] durchgeführt. Alternativ wäre auch ein Wavelet-Transformation [32] möglich

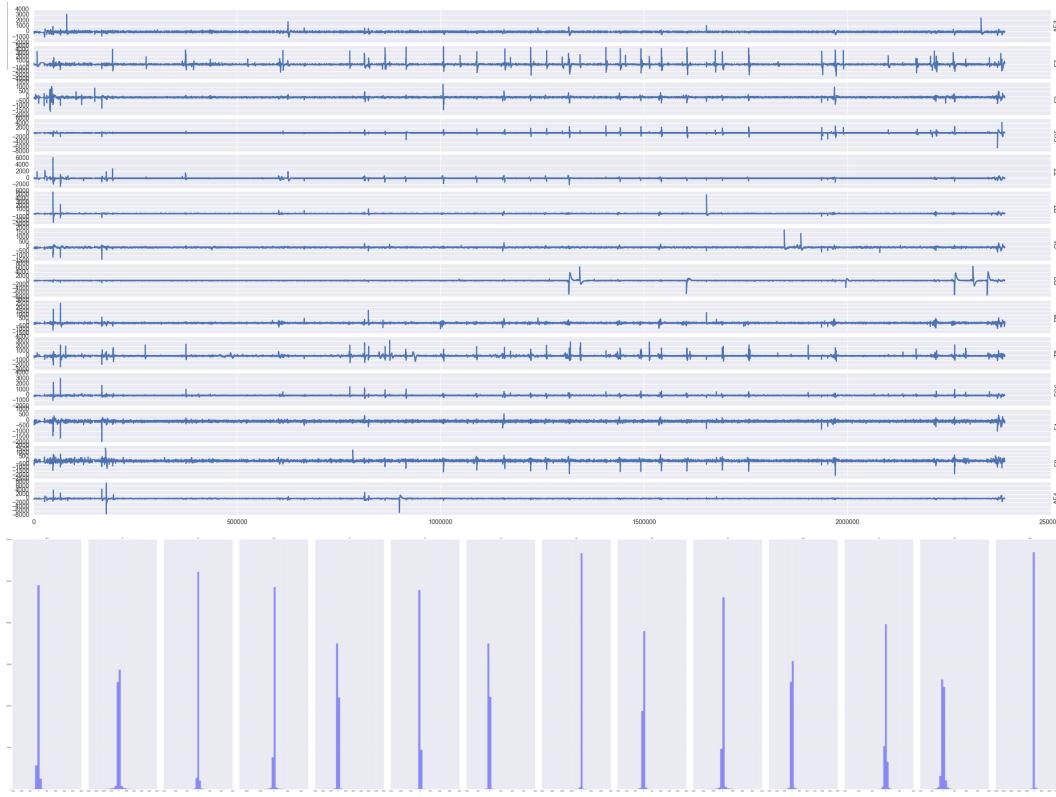


Abb. 5.6: Die Rohsignale pro Sensore und das dazugehörige Histogramm. Die Werte verteilen sich zwischen -8000 bis 8000, ein deutliches Maximum im Histogramm ist zwischen -100 und 100 erkennbar.

gewesen. Beide Ansätze fanden sich in der Literatur Recherche, aus Zeitgründen wurde auf einen Vergleich der Ansätze verzichtet. Die Fourier-Transformation bildet ein aperiodische, kontinuierliches Signal auf ein diskretes, periodisches Frequenzspektrum ab. Um die Frequenzbänder zu erhalten, wird das Ergebnis auf die gewünschten Frequenzen reduziert (Abb. 5.7).

5.3 Merkmalsextraktion

Nach der Aufbereitung und Verarbeitung der EEG-Signale, müssen nun signifikante Daten extrahiert und eine Merkmalsmenge erstellt werden, welche vom Klassifikator erkannt werden kann.

Aus der validierten Hypothese aus dem Experiment (vgl. Kapitel 4.2) folgt, dass es eine Veränderung vom Anfang zum Ende der Testfahrt geben muss. Um den Effekt zu verstärken, wurden Datensätze aus den Minuten 5-10 (Wach) und 20-25 (Müde) entnommen und jeweils verglichen. Es wurde eine fließende Veränderung

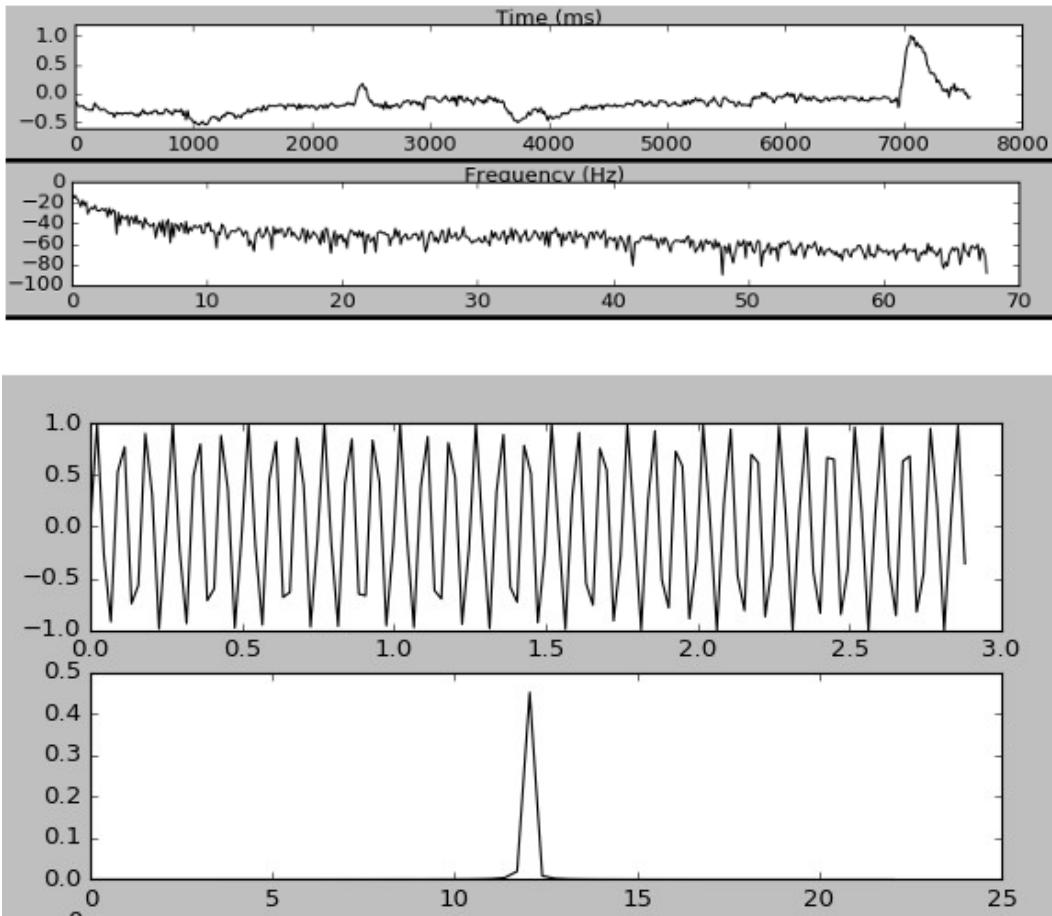


Abb. 5.7: Oben: EEG Signal darunter das Frequenzspektrum. Unten: die selbe Anordnung für ein Tonsignal und beschreibt eine Sinusschwingung mit 12KHz. Nach der FFT zeigt sich ein Maximum bei dieser Frequenz.

angenommen, welche unter Umständen schwerere erkennbar wäre. Die beiden Datensätze wurden durch die Verarbeitungskette geschickt und in einer Testmenge gespeichert.

Wie im vorherigen Kapitel gezeigt, lassen Veränderungen in den Frequenzbändern verschiedene Schlüsse zu. So konnten Pal et al. [33] Zusammenhänge zwischen Veränderungen der Alpha- und Theta-Wellen und Veränderungen der kognitiven Fähigkeiten feststellen. Dies wurde bei der Analyse der Daten berücksichtigt und besonders auf eine Veränderung der Theta- und Alpha-Wellen geachtet.

Mehrere Merkmale wurden testweise extrahiert und manuell auf Unterschiede geprüft. Werte aus der Spracherkennung wie Nulldurchgangsrate und Signalernergie wurden sowohl für das verarbeitete Signal, als auch für die Frequenzbänder durchgeführt. Die Ergebnisse lieferten keine eindeutigen Unterschiede. Statistische

Merkmale, wie die Standardabweichung und die Varianz waren ebenfalls nicht eindeutig.

Kleinere Abweichungen in der Ausschlagshöhe der Frequenzbänder konnten über die gesamten Datensätze „Wach“ und „Müde“ erkannt werden. Am deutlichsten waren die Veränderungen in den Theta-Wellen der Sensoren „AF3“, „AF4“, „F3“, „F4“, „F7“, „F8“ erkennbar. Um mögliche Fehlerquellen zu verringern und die Erkennung zu vereinfachen, wurden Sequenzen mit größeren Ausreißern manuell bereinigt. Da sich die Unterschiede in der Ausschlagshöhe nur auf die gesamte Zeitdauer von 5min erstreckten und nicht auf ein Zeitfenster von 1s, war zu prüfen, ob der Klassifikator in der Lage ist, die beiden Zustände sicher zu erkennen.

5.4 Klassifikator

Nachdem im vorherigen Abschnitt eine passende Merkmalsmenge erstellt wurde, geht es nun um die Entscheidung, ob der Fahrer „Müde“ oder „Wach“ ist bzw. ob das System eine Müdigkeitsmeldung erscheinen lässt. Für diese Klassifizierung werden im allgemeinen Machine-Learning-Algorithmen verwendet. Anhand von markierten Datensätzen wird versucht den Algorithmus zu trainieren (Überwachtes Lernen). Dies dient dem Ziel, dass er auch unbekannte Daten klassifizieren kann. Dieser Vorgang wird Generalisierung bezeichnet und ist auch im menschlichen Lernen ein wichtiger Schritt. Für die Anwendung wurde zur Klassifizierung ein künstliches Neuronales Netz (KNN) ausgewählt. Es basiert auf einem erweiterten Perceptron / McCulloch-Pitts-Neuron [34] und ist der Funktionsweise des menschlichen Gehirns bzw. seinen Neuronen nachempfunden [35]. Ein KNN lässt sich im einfachsten Fall durch eine Merkmalsmenge $X = x_1, x_2 \dots x_n$, dazugehörige Gewichte $W = w_1, w_2 \dots w_n$, eine Übertragungsfunktion \sum und eine Schwellwertfunktion θ beschreiben (Abb. 5.8).

Dieser Aufbau kann schon einfache Aufgaben, wie bspw. ein logisches „UND“, lösen. Jedoch lässt sich schon ein logisches „XOR“ nicht mehr abbilden. Dafür müssen weitere Schichten von Neuronen (Hidden Layers) hintereinander geschaltet werden - das sog. Multi Layer Perceptron (MLP, Abb. 5.9).

Es existiert kein bekannter Algorithmus für eine allgemeine Wahl der optimalen initialen Parameter eines KNNs (Gewichte, Anzahl der Hidden Layers). Diese werden initial entweder zufällig gesetzt oder durch Ausprobieren gefunden. Vuckovic et al. [17] hatten sich mit diesem Thema genauer beschäftigt.

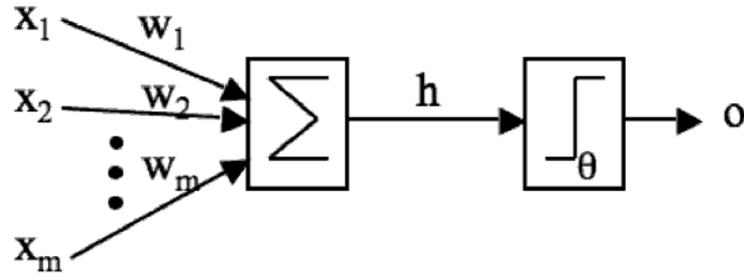


Abb. 5.8: Darstellung eines McCulloch-Pitts-Neurons. Die Merkmale X werden mit den Gewichten W multipliziert und in \sum summiert. Wenn $h > \theta$ „feuert“ das Neuron ($o = 1$) [35].

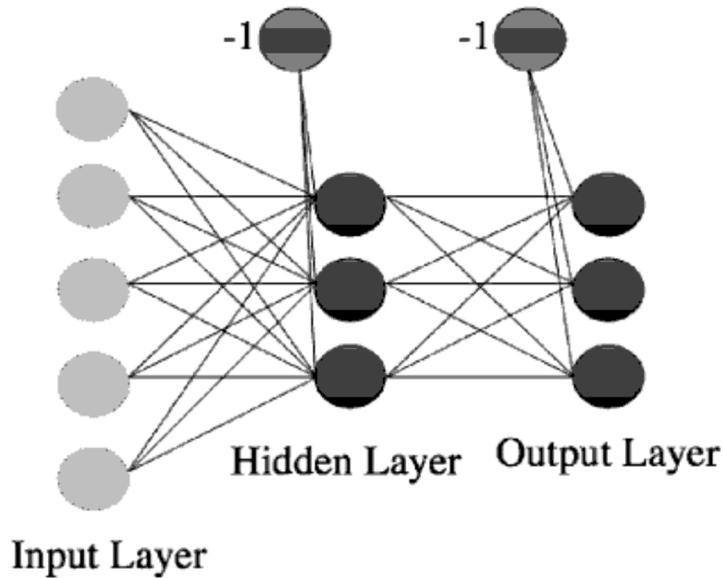


Abb. 5.9: Darstellung eines Neuronalen Netzes mit mehreren Schichten (Multi Layer Perceptron, MLP) [35].

Die Merkmalsvektoren kommen zu gleichen Teilen aus der „Wach“ und „Müde“-Menge und sind mit den jeweiligen Klassen 0 und 1 versehen. Insgesamt wurden 1170 Datensätze mit jeweils 24 Werten pro Vektor eingesetzt. Vor dem Training wurde die Merkmalsmenge in Trainings- und Testmenge (2:1) aufgeteilt. Intern wird beim Training noch einmal ein Teil der Daten zum validieren genutzt (15%). So wird sichergestellt, dass die Tests nie mit Daten durchgeführt werden, die schon beim Training genutzt wurden. Sonst kann es zum sog. Overfitting kommen, das bedeutet, dass das Netz zu genau auf die Trainingsdaten eingestellt ist und nicht mehr generalisiert.

Für die Anwendung wurde das Training des KNN mit verschiedenen Parametern durchgeführt. Die Menge der Eingabevektoren (X) ist durch die Größe des Merkmalsvektors vorgegeben. Da lediglich zwei Klassen (Wach und Müde) existieren,

reicht ein binärer Ausgabevektor aus. Die Frage nach der Menge der Hidden Layer lässt sich nicht per se beantworten. In Versuchen ergaben vier Schichten die besten Ergebnisse. Weitere Parameter können beim Training eingestellt werden, ein variabler Trägheitsterm (Momentum) verhindert bspw. das Feststecken in lokalen Minima, die Lernrate steuert die Lerngeschwindigkeit im Vergleich zur Genauigkeit. Gute Ergebnisse ließen sich mit einem Momentum 0,25 und einer Lernrate von 0.005 erzielen. Das Training ist beendet, wenn die Gewichte und die Fehlerrate des KNN konvergieren (das kann theoretisch nie passieren) oder die maximale Iterationen erreicht wurde. Für das Training wurde diese auf 5000 eingestellt und ein Training dauerte maximal 30Min. Höhere Iterationszahlen führten nicht zu einer Verbesserung der Erkennung. Das Ergebnis ist aufgrund der zufälligen Anfangsparameter immer unterschiedlich, so wurde das Training vier Mal parallel in verschiedenen Threads durchgeführt. Bis zum aktuellen Ergebnis wurden ca. 200 Trainingsläufe gestartet.

Die Testmenge wird nun in das KNN eingegeben und überprüft, ob die richtige Klasse erkannt wird. Daraus ergibt sich eine Ergebnis-Matrix (Tab. 5.1), in der jede richtig und falsch erkannte Klassifizierung ersichtlich wird. Die Diagonale zeigt die richtig erkannten Klassen. Die Erkennungsrate pro Klasse befindet sich in der letzten Spalte. Da beide Klassen zu gleichen Teilen vorhanden sind, ist die Gesamterkennungsrate das Mittel der beiden Einzelraten.

Tab. 5.1: Die Tabelle zeigt das Klassifizierungsergebnis der Test- und Trainingsdaten

	Wach	Müde	Erkannt
Wach	357	227	61%
Müde	217	367	62%
Gesamt	-	-	61.5%

6 Ergebnis

In der vorgestellten Arbeit wurde eine Anwendung zur Müdigkeitserkennung entwickelt. Sie liest Rohdaten des Emotiv EEG-Headsets ein, verarbeitet und klassifiziert sie. Wird Müdigkeit erkannt, wird dies dem Fahrer mitgeteilt. Die Datenerhebung ist lose gekoppelt und kann die EEG-Daten über mehrere Wege übertragen. Für die Verarbeitung der Daten stehen mehrere Klassen zur Verfügung, die zu einer Verarbeitungskette verbunden werden. Für die Klassifizierung wurde ein KNN eingesetzt, welches performant die EEG-Sequenzen einteilt. Die Anwendung fügt sich in die bestehende Simulationsumgebung der Reutlingen University ein. Vier Testfahrten für Trainingsdaten wurden im Fahrsimulator der Hochschule Reutlingen samt Videobild aufgenommen.

Präzision und Genauigkeit (1) konnten nicht wie erwartet umgesetzt werden. Die Erkennungsrate liegt derzeit bei ca. 61,5% - das ist für ein sicherheitsrelevantes System deutlich zu niedrig. Fehlertoleranz (2) und Fehlerbehandlung (3) wurden umgesetzt und durch Unit- und Integrationstests abgesichert.

Die Verarbeitungsgeschwindigkeit (4) der empfangenen Daten beträgt während eines Benchmarktests im Schnitt 200ms pro Sequenz und liegt damit deutlich unter der Liefergeschwindigkeit des EEGs (Sequenz \equiv 128 Werte \equiv 1000ms). Die Portierung des Systems ist theoretisch sehr einfach möglich, wenn während der Fahrt ein Laptop genutzt wird (5). Die Handhabung und der Komfort des Headsets sind im Vergleich zu medizinischen EEGs deutlich verbessert (6). Das EEG lässt sich ähnlich wie eine Mütze aufsetzen, jedoch ist die Einrichtung der Sensoren aufwändiger als erwartet (beste Signal-Qualität auf allen Sensoren). Die kabellose Übertragung sorgt für maximale Beweglichkeit, dennoch ist das Headset für den Produktiveinsatz eher ungeeignet.

Tests auf einem Smartphone oder Tablet müssen gesondert erfolgen, da es unter anderem von den Treibern des EEG Headsets abhängt (7). Lasttests wurden nur auf einem Laptop (Lenovo ThinkPad W530) durchgeführt (8). Der Einbau in die Simulationsumgebung ist bis zum Eintragen der Werte im CAN-Bus umgesetzt. Das Herausnehmen der Werte ist noch nicht vollständig umgesetzt, da bisher die Integration der CarInterface Anwendung nicht funktionierte (9). Die PoSDBoS Anwendung lässt sich sehr gut auf verteilten Systemen ausführen, auch über den Anwendungsfall des Fahrsimulators hinaus. Die akquirierten Daten können via http

an die Verarbeitungsschicht übertragen werden. Auch das Verschicken des Ergebnisses der Klassifizierung per http wäre denkbar und einfach umzusetzen (10). Die Art der Benachrichtigung über eine erkannte Müdigkeit ist noch nicht vollständig umgesetzt. Derzeit wird dem Fahrer entweder ein grüner oder roter Bildschirm angezeigt.

Die komplette Anwendung ist mit Docstring¹ versehen, aus denen eine html-Dokumentation erzeugt werden kann. Weiterhin sind schwierige Code-Teile mit einfachen Beispielen erweitert, um den Einstieg zu erleichtern. Ob es so möglich ist, als Neuling, selbständig die Anwendung zu verstehen bzw. zu erweitern hängt wohl auch von den jeweiligen Grundkenntnissen ab. Werden jedoch Veränderungen vorgenommen, kann durch die Unit-Tests sichergestellt werden, dass die Klassen weiterhin das Richtige tun. Integrationstests sind nur teilweise umgesetzt.

¹<https://www.python.org/dev/peps/pep-0257/#what-is-a-docstring>

7 Fazit und Ausblick

Die Ergebnisse zeigen eine vielversprechende Grundlage und bilden das Grundgerüst für weitere Arbeiten. Die passende Architektur und Infrastruktur ist vorbereitet. Die Annahmen der Experiment-Hypothese wurde bestätigt, wenngleich sie sich nicht im erwarteten Maße in den EEG Daten niedergeschlagen hatte.

Die unzureichende Genauigkeit ist der wichtigste Anhaltspunkt der weiteren Forschung. Es werden derzeit lediglich zwei von drei Sequenzen aus den besten Daten richtig erkannt. Die Gründe hierfür lassen sich nicht eindeutig klären. Es könnte bei der Datenakquise - dem Experiment - beginnen, an Fehlern oder falscher Berechnungen während der Verarbeitung oder ungünstigen Startparametern beim Training des KNN liegen. Das Experiment hatte, bei Betrachtung objektiver Merkmale, sowie subjektiver Einschätzung der Testpersonen selbst, den gewünschten Effekt. Für weitere Schritte könnten die Experimente wiederholt werden und ggf. über einen längeren Zeitraum (~1h) laufen. So wäre der Unterschied zwischen Wach und Müde unter Umständen deutlicher erkennbar. Auch die Aufnahme von Testdaten in verschiedenen Szenarien (abwechslungsreiche Tagfahrt und reizarme Nachtfahrt) und unterschiedlichen Startparametern (bspw. Schlafmenge vor dem Experiment), könnten die Veränderung deutlicher machen. Um Fehler beim Headset und den gelieferten Daten auszuschließen, müssen zuvor allerdings die Daten des Headsets und der genutzten Emokit Bibliothek noch einmal validiert werden. Die Berechnungen der Verarbeitungslogik könnten mit etablierten EEG-Bibliotheken verglichen und werden.

8 Master Projekt

Neben der Entwicklung der Anwendung zur Müdigkeitserkennung, standen einige Termine für das IoT-Lab an. Unter Anderem war das Thema auf der WVK.15¹, der InformaticsInside², am Tag der offenen Tür und dem Studentag präsent. Die in diesem Dokument genutzte LATEX-Vorlage für die Ausarbeitung³ wurde an den Stand der vorgegebenen Word-Vorlage angepasst und zur Verfügung gestellt.

Die Einarbeitung und vor allem die Dokumentation des Fahrsimulators war eine weitere wichtige Aufgabe. Viele Fragen wurden nach dem Weggang von Emre Yay nicht abschließend geklärt. So wurde viel Zeit mit Suchen, Debuggen und Nachfragen benötigt. Um Anderen diesen Aufwand zu ersparen war es wichtig, den Simulator zu dokumentieren. Dazu musste zuerst das Format entschieden werden, da mehrere Personen, am besten auch parallel, an der Dokumentation arbeiten sollten. Ein Wiki⁴ in der Umgebung der IoT-Website erschien am besten für diese Aufgabe. Die Struktur und initialer Inhalt waren die ersten Aufgaben für die Dokumentation. Weiterhin wurden Tutorials erstellt, sodass ein Neuling einen Einstiegspunkt für einfache Aufgaben (bspw. Simulation starten) vorfindet. Juniors im IoT-Lab konnten davon bereits profitieren.

Während der Entwicklung gab es immer wieder Kommunikationsbedarf mit verschiedenen Supportstellen. So traten Probleme mit OpenDS, CANoe und Emokit auf, die mit dem jeweiligen Support per Email oder Telefon geklärt werden mussten. Den meisten Aufwand erzeugte in dieser Hinsicht das Einschicken des EEG Headsets. Nachdem der Defekt eines Sensors Anfang Februar festgestellt wurde, war nach einigen Emails mit dem Emotiv Support und in Absprache mit MKI Service klar, dass das Headset eingeschickt werden musste. Dies passierte Anfang März und verzögerte sich nochmal wegen Problemen am Zoll. Ende April kam das Headeset dann im TechCenter der Firma Emotiv an. Mehrere Anfragen bei Emotiv lieferten leider keine neuen Erkenntnisse. Ende Mai kam die Nachricht, dass der Fehler gefunden und behoben wäre. Der beschriebene Fehler wurde jedoch bereits in Reutlingen überprüft. Dies wurde Emotiv mitgeteilt mit der Bitte noch

¹<http://wvk.reutlingen-university.de/index.php?site=topic&id=150299>

²<http://www.infoinside.reutlingen-university.de/index.php?site=program>

³<https://relax.reutlingen-university.de/course/view.php?id=6884>

⁴http://iotlab.reutlingen-university.de/iotlab_wiki/index.php/Driving_Simulator

einmal zu testen. Beim zweiten Test wurde dann festgestellt, dass das Headset nicht stabil laufe und es wohl ein anderes Problem sei. Nach weiteren drei Wochen entschied Emotiv sich, ein neues Gerät zu verschicken. Weitere Nachfragen erreichten dann auch die Herausgabe eines Tracking-Codes, sodass der Sendestatus des EEGs verfolgt werden konnte. Die Notwendigkeit des häufigen Nachfragens und die Reaktionszeiten von jeweils mindestens einer Woche, waren frustrierend und zeitraubend.

9 Literaturverzeichnis

- [1] Strategy Analytics. Advanced driver assistance systems forecast - aug 2015. <https://www.strategyanalytics.com/access-services/automotive/powertrain-body-chassis-and-safety/market-data/report-detail/advanced-driver-assistance-systems-forecast---aug-2015>, 2015. Zugriff: 2016-08-31.
- [2] X. Mosquet, M. Andersen, and A. Arora. A roadmap to safer driving through advanced driver assistance systems. <https://www.bcgperspectives.com/Images/MEMA-BCG-A-Roadmap-to-Safer-Driving-Sep-2015.pdf>, 2015. Zugriff: 2016-08-31.
- [3] Daimler AG. Attention assist, 2013. Available at <https://blog.daimler.de/2013/09/04/einfach-technik-der-attention-assist-von-mercedes-benz>, Zugriff: 2016-08-31.
- [4] Stat. Bundesamt. Zahl der Verkehrstoten steigt im Jahr 2015 vorrausichtlich auf etwa 3450. https://www.destatis.de/DE/PresseService/Presse/Pressemitteilungen/2015/12/PD15_463_46241.html, 2015. Zugriff: 2016-08-31.
- [5] C. Evers. Unterschätzte Risikofaktoren Übermüdung und Ablenkung als Ursachen für schwere Lkw-Unfälle. http://www.dvr.de/presse/seminare/904_20.htm, 2008. Zugriff: 2016-08-31.
- [6] H. H. Jasper. The ten twenty electrode system of the international federation. *Electroencephalography and Clinical Neurophysiology*, 10:371–375, 1958.
- [7] G. Deuschl and A. Eisen. IFCN guidelines for topographic and frequency analysis of EEGs and EPs. 1999.
- [8] Robert Bosch GmbH. Bosch Driver Drowsiness Detection, 2012. Available at <http://www.bosch-presse.de/pressportal/en/bosch-driver-drowsiness-detection-41616.html>, Zugriff: 2016-08-31.

- [9] A. Eskandarian and A. Mortazavi. Evaluation of a smart algorithm for commercial vehicle driver drowsiness detection. In *2007 IEEE Intelligent Vehicles Symposium*, pages 553–559, June 2007.
- [10] L. Zhang, F. Liu, and J. Tang. Real-Time System for Driver Fatigue Detection by RGB-D Camera. *ACM Trans. Intell. Syst. Technol.*, 6(2):22:1–22:17, March 2015.
- [11] L. M. Bergasa, J. Nuevo, M. A. Sotelo, R. Barea, and M.E. Lopez. Real-time system for monitoring driver vigilance. *Intelligent Transportation Systems, IEEE Transactions on*, 7(1):63–77, March 2006.
- [12] J. Vicente, P. Laguna, A. Bartra, and R. Bailon. Detection of driver's drowsiness by means of HRV analysis. In *Computing in Cardiology, 2011*, pages 89–92, Sept 2011.
- [13] E. Rogado, J.L. Garcia, R. Barea, L. M. Bergasa, and E. Lopez. Driver fatigue detection system. In *Robotics and Biomimetics, 2008. ROBIO 2008. IEEE International Conference on*, pages 1105–1110, Feb 2009.
- [14] R. N. Khushaba, S. Kodagoda, S. Lal, and G. Dissanayake. Driver Drowsiness Classification Using Fuzzy Wavelet-Packet-Based Feature-Extraction Algorithm. *Biomedical Engineering, IEEE Transactions on*, 58(1):121–131, Jan 2011.
- [15] R. R. Johnson, D. P. Popovic, R. E. Olmstead, M. Stikic, D. J. Levendowski, and C. Berka. Drowsiness/alertness algorithm development and validation using synchronized EEG and cognitive performance to individualize a generalized model. *Biological psychology*, 87(2):241–250, May 2011.
- [16] A. Subasi. Automatic Recognition of Alertness Level from EEG by Using Neural Network and Wavelet Coefficients. *Expert Syst. Appl.*, 28(4):701–711, May 2005.
- [17] A. Vuckovic, V. Radivojevic, A. C.N. Chen, and D. Popovic. Automatic recognition of alertness and drowsiness from EEG by an artificial neural network. *Medical Engineering & Physics*, 24(5):349 – 360, 2002.
- [18] R. S. Huang, K. J. Chung, L.-L. Tsai, and O. T. C. Chen. Eeg pattern recognition-arousal states detection and classification. In *Neural Networks, 1996., IEEE International Conference on*, volume 2, pages 641–646 vol.2, Jun 1996.

- [19] C.-T. Lin, R.-C. Wu, S.-F. Liang, W.-H. Chao, Y.-J. Chen, and T.-P. Jung. EEG-based drowsiness estimation for safety driving using independent component analysis. *IEEE Trans. Circuits Syst. I, Reg. Papers*, pages 2726–2738, 2005.
- [20] B. J. Wilson and T. D. Bracewell. Alertness monitor using neural networks for EEG analysis. In *Neural Networks for Signal Processing X, 2000. Proceedings of the 2000 IEEE Signal Processing Society Workshop*, volume 2, pages 814–820 vol.2, 2000.
- [21] K. B. Khalifa, M. H. Bedoui, R. Raytchev, and M. Dogui. A portable device for alertness detection. In *Microtechnologies in Medicine and Biology, 1st Annual International Conference On. 2000*, pages 584–586, 2000.
- [22] H. Park, S. Oh, and M. Hahn. Drowsy Driving Detection Based on Human Pulse Wave by Photoplethysmography Signal Processing. In *Proceedings of the 3rd International Universal Communication Symposium, IUCS '09*, pages 89–92, New York, NY, USA, 2009. ACM.
- [23] A. Zhang and F. Liu. Drowsiness detection based on wavelet analysis of ECG and pulse signals. In *Biomedical Engineering and Informatics (BMEI), 2012 5th International Conference on*, pages 491–495, Oct 2012.
- [24] J. Engstrom, E. Johansson, and J. Ostlund. Effects of visual and cognitive load in real and simulated motorway driving. *Transportation Research Part F: Traffic Psychology and Behaviour*, 8(2):97–120, March 2005.
- [25] J. Horne and L. Reyner. Vehicle accidents related to sleep: a review. *Occupational and Environmental Medicine*, pages 289–294, May 1999.
- [26] M. Amd. How to get a alpha , beta , gamma, delta, theta waves from raw wave(eeg). https://www.researchgate.net/post/How_to_get_a_alpha_beta_gamma_delta_theta_waves_from_Raw_waveEEG, 2016. Zugriff: 2016-08-31.
- [27] S. Butterworth. On the theory of filter amplifiers. *Wireless Engineer*, 7, 1930.
- [28] S. Lei and M. Roetting. Influence of Task Combination on EEG Spectrum Modulation for Driver Workload Estimation. *Human Factors*, 53(2):168–179, 2011.

- [29] J.-Y. Lv, T. Wang, J. Qiu, S.-H. Feng, S. Tu, and D.-T. Wei. The electrophysiological effect of working memory load on involuntary attention in an auditory–visual distraction paradigm: an ERP study. *Experimental Brain Research*, 205(1):81–86, 2010.
- [30] A. Gundel and G. F. Wilson. Topographical changes in the ongoing EEG related to the difficulty of mental tasks. *Brain Topography*, 5(1):17–25, 1992.
- [31] S. Bochner and K. Chandrasekharan. *Fourier Transforms*. Princeton University Press, 1949.
- [32] C. K. Chui. *An Introduction to Wavelets*. Academic Press Professional, Inc., San Diego, CA, USA, 1992.
- [33] N. R. Pal, C.-Y. Chuang, L.-W. Ko, C.-F. Chao, T.-P. Jung, S.F. Liang, and C.-T. Lin. EEG-based Subject- and Session-independent Drowsiness Detection: An Unsupervised Approach. *EURASIP J. Adv. Signal Process*, 2008:192:1–192:11, January 2008.
- [34] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.
- [35] S. Marsland. *Machine learning : an algorithmic perspective*. Chapman & Hall/CRC machine learning & pattern recognition series. CRC Press, Boca Raton, 2009. A Chapman & Hall book.

10 Eidesstattliche Erklärung

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Arbeit selbstständig erstellt, keine nicht genannte fremde Hilfe in Anspruch genommen und alle von mir verwendeten Hilfsmittel und Quellen in der Arbeit benannt und kenntlich gemacht habe.

Mir ist bekannt, dass eine unwahre Erklärung als Täuschung gewertet wird.

Ort,

Datum

Unterschrift (Vor- und Nachname)