Driver Drowsiness Detection System

Dakota Mouton, Ali Imran, Coady Lewis

CONCEPT OF OPERATIONS

REVISION – I
5 December 2021

Concept of Operations
for
Driver Drowsiness Detection System

Team 61

Approved by:

_____
Project Leader Date

_____
Prof. Kalafatis Date

_____
T/A Date

Change Record

Rev.	Date	Originator	Approvals	Description
-	9/16/2021	Entire Team		Initial Submission
I	12/5/2021	Coady Lewis		Original MCU replaced by Raspberry Pi; Impact Updated

Table of Contents

List of Figures

1. Executive Summary

Fatigue has proven to be a significant cause of driver accidents. This can be a major problem for industries, such as trucking, that rely on people driving for extended periods of time. Currently, trucking insurance companies enforce a daily limit on how long a trucker can drive. However, this does not guarantee that the drivers are not driving fatigued. Our goal is to provide a system that will detect fatigue and notify the user that they are falling asleep. The Driver Drowsiness Detection System will accomplish this with an electroencephalogram (EEG) that will alert the driver when it detects a state of fatigue. The DDDS will accurately report the fatigue level of the user and quickly inform them when they begin to get tired. This will provide trucking insurance companies necessary information to keep their drivers safe.

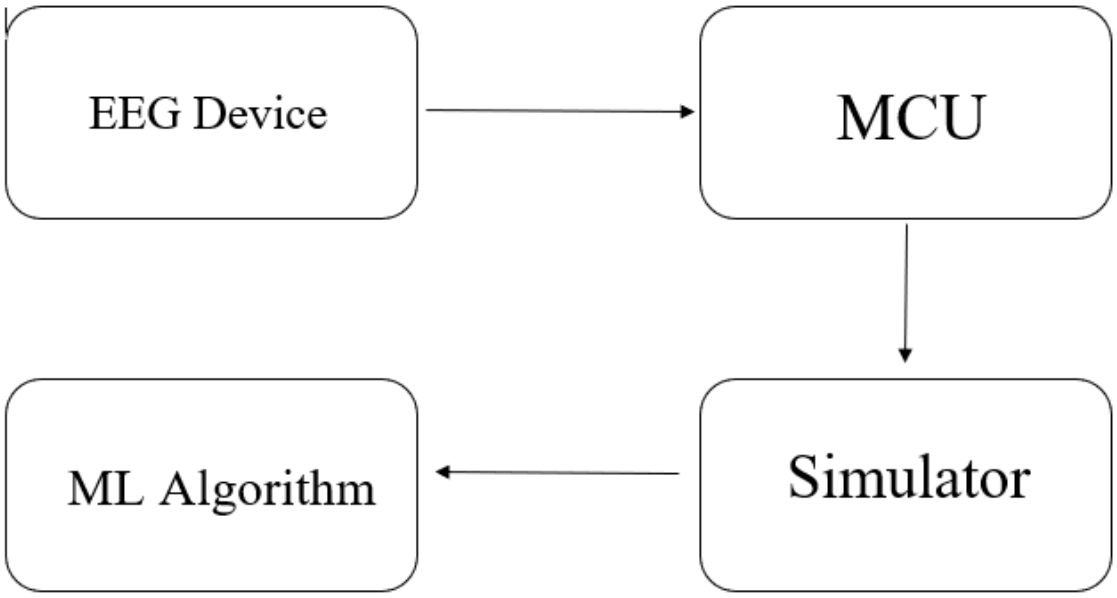


Figure 1: Driver Drowsiness Detection System Block Diagram

2. Introduction

This document is an introduction to the Driver Drowsiness Detection System. This system is capable of providing up to date information about the user's drowsiness, as well as informing the user when it detects a decrease in their level of alertness. The DDDS will serve as a tool to keep drivers alert while driving, as well as allowing insurance companies to monitor the mental state of their drivers.

2.1. Background

The current method trucking insurance companies employ to monitor their drivers alertness is an electronic log device (ELD). These electronic log devices automatically record driving time as well as hours of service. The problem with this method is that while the ELD accurately tracks the driving time, it cannot track sleeping patterns or other indicators of a driver's mental fatigue. This means that a driver can seemingly follow all the rules set by their insurance company, or laws set by the government, and still pose a serious risk to other drivers on the road as well as themselves and their cargo. The DDDS solves this problem by directly monitoring the driver's fatigue. This way it is unnecessary to monitor factors such as total driving time which are only possible indicators of mental state. Since the limiting of driving time is done as an attempt to ensure alert driving, insurance companies could potentially remove these limits and refer only to the DDDS to accomplish the same thing. This could increase the overall driving capacity of the truck drivers, since now the drivers would only be limited by their own fatigue as opposed to time.

2.2. Overview

Our system will be used to monitor brain waves to detect an increase in fatigue. We will detect these brain waves using an EEG device. The information captured by the EEG device will then be fed through an analog circuit designed to cancel out excess noise. A microcontroller will then process this data so it can be interpreted by a machine learning algorithm. Once the machine learning algorithm collects enough data it will be able to detect when a driver begins to feel fatigued, and how fatigued they are. After the algorithm detects increasing levels of fatigue it will notify an app that will alert the driver as well as the insurance company.

2.3. Referenced Documents and Standards

- IEC 6061 Technical standard for medical electrical equipment
- IEEE Recommended Practice for Neurofeedback Systems
- https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3571819/

3. Operating Concept

3.1. Scope

The Driver Drowsiness Detection System will provide both trucking companies and insurance companies with a way to measure fatigue for each individual driver. The system will collect EEG signals from a band on the driver's head, and these signals will be sent to a microcontroller to be processed. The ML algorithm will determine the driver's state of awakeness by comparing the signals with a calibration signal. The calibration signal will be individual to each user. When a driver is in a fatigued state, the system will alert the driver via an app and advise them to pull over. The app will also log each of these instances and store the data for future reference. Companies will be able to use this data as an individualized metric for the fatigue of drivers. This could eventually be a more reliable safety measure than the common 8 hour rule. A simulator will be constructed for the collection of realistic data in a controlled and safe environment.

3.2. Operational Description and Constraints

The Driver Drowsiness Detection System and the data it collects will be designed for use by long distance truck drivers, their employers, and their insurance companies. It will collect data from an EEG module on the driver's head, analyze the data, and output warnings if the driver is in a fatigued state.

Constraints

- Each driver must complete a calibration procedure to maximize the accuracy of the system.
- The EEG band must be worn securely on the head at all times when the system is in use.
- The EEG module will not be waterproof. It will not function properly if the driver is profusely sweating or in the rain.
- The system will not function well in extreme heat.
- The system must be connected to a smartphone or laptop to use the datalogging feature.
- The budget for the system is $400.

3.3. System Description

- EEG: This subsystem will include electrodes that directly read potential values from several points on the driver's head. The final iteration will have the electrodes fixed in the band of a hat. Each channel will be amplified, filtered, and fed directly to one of the analog inputs of the microcontroller. The filtering circuits will be fine tuned in testing to make the signals more usable.
- Microcontroller: The controller will convert each analog EEG input to a digital signal. It will be connected to a bluetooth module to output the data to either a laptop, or the application running on a smartphone. As of 12/5/21, a Raspberry Pi will replace the function of the microcontroller subsystem.
- ML Algorithm: This algorithm will determine the current state of the driver's fatigue. It will compare the live EEG signals with a calibration signal, and it will update the

tolerances for a match as more data is tested. The calibration waveforms will also be adjusted after many trials with feedback on the simulator.

- Application: The application will be interfaced with the other subsystems and will serve as the output of the entire system. It will display the current state of awakeness, and it will issue visible and audible warnings when the driver is in a fatigued state. It will also compile data on the number of times the driver registers as fatigued or critically fatigued, as well as the time and duration of each instance. This data can be sent to trucking agencies or insurance companies.

3.4. Modes of Operations

Active Operation - The drowsiness detection system will always be in operation when the device is turned on and will be continuously monitoring the brainwaves of the user to determine their current level of awakeness.The device will monitor for these three states of awakeness and will send an alert message to the user when a specific level is reached.

- **Awake -** When the user has what is considered normal brainwave activity being associated with regular awareness, the device will not send a message to the user but will look for changes in their brain activity. This will be it's idle state of operation.
- **Drowsy -** When the device detects the level of brain activity between the normal brain activity threshold and the critical level threshold, the device will send a warning message to the user about their current state and will suggest coming to a point of rest or a break to bring their levels back into the awake state.
- **Critical Drowsy -** When the device detects the level of brain activity that is below the drowsy threshold, the device will send an alert to the user telling them about their current state and that they need to rest as soon as possible to bring their awakeness back to a safe level. When this level is reached, the insurance company using this device on the driver will also be alerted of the current state of the driver as documentation for potentially negligent driving.

3.5. Users

Drivers - The truck drivers will use this device in their trucker hats or a baseball cap to monitor their levels of awakeness. They will require little training, only requiring knowledge of how to turn on the device and properly set it on their head with little obstruction, as well as what the alerts mean from the device about their current levels and what to do when those levels are reached. The device will sit on their head inside their hat and will continuously monitor their brain activity.

Insurance Companies - The agents of the truck drivers will need to understand what the levels of awakeness means for the drivers current state, what the driver should do when they reach those alerts and how the driver should properly wear the device to ensure it works properly. The agent can use the data collected from the drivers to determine how long and how quickly it takes drivers to fatigue after continuous driving to determine their coverage rates and to protect themselves from reliability when the driver was being negligent and unsafe with their driving. The insurance companies will benefit from the data

collected and this will increase the safety of both the truck drivers and the general public from a potentially catastrophic event from the driver losing control due to their fatigue.

3.6. Support

User Manuals - The driver drowsiness detection system will come with a user manual that will cover how to power and operate the device, safety warnings and instructions, how levels of awakeness are determined, what the levels of awakeness mean and suggestions for what to do when those levels are reached. There will also be simple troubleshooting solutions included to try and help alleviate some of the more basic issues one might run into when using an electronic device.

4. Scenarios

4.1. Truck Driving

The main use case for the Driver Drowsiness Detection System will be with truck drivers. Truck drivers spend a lot of hours out on the road, so they are one of the biggest groups at risk of being drowsy while operating a vehicle. Also, truck drivers normally drive large vehicles at highway speeds which may end up causing catastrophic wrecks. The idea is to have the operators wear a hat which includes the EEG device and microcontroller. The EEG device will record the brain waves of the operator and send that information to the microcontroller. The microcontroller would then process the data and use a program with the constructed machine learning algorithm to show an output describing the state of the driver. This output and the brain wave data could then be sent to the operator's phone, tablet, or laptop which is then sent to the respective insurance companies. Once the driver is detected to be drowsy, an alert will be made on their device to notify them to pull their vehicle over.

4.2. Long Commutes

The DDDS could potentially be used for general use. People that commonly take long road trips but are afraid of falling asleep at the wheel could use our device to monitor their fatigue. Similarly, their phone could be used to alert them when fatigue has been detected, and they could pull over to rest.

5. Analysis

5.1. Summary of Proposed Improvements

- Since the DDDS is implemented in a portable hat, it can be easily carried around and does not require any sort of heavy installation within the truck itself. It is also non-intrusive and does not look out of place.
- The machine learning algorithm will allow fatigue to be reliably detected based on the kinds of brain signals being read with very few false positives.

- The application can record the fatigue data output and possibly be sent to insurance companies and the trucking agency.
- With the data from the DDDS, truck drivers could make better informed decisions on the routes that they end up taking. They could take shorter or longer routes based on when they experience fatigue. They could also plan breaks and rests accordingly.
- With the data from the DDDS, insurance companies could change their coverage rates and protect themselves when drivers have shown they have been negligent to their fatigue alerts when an accident occurs.

5.2. Disadvantages and Limitations

- The EEG device will not be perfectly accurate in collecting brain wave data due to possible external interference and internal interference from other brain waves.
- Due to budget constraints, the EEG sensors will be of limited quantity and channels. This may lead to noisy outputs from the sensors.
- The brain is a complex organ, so extracting data from a small sample may not 100% accurately reflect the expected brain activity of others.
- The machine learning algorithm will be built from limited data collected from the virtual simulation and possibly from available online datasets.

5.3. Alternatives

- One alternative solution is using a visual detector. The detector would record the facial expressions and eyes of the vehicle operator to check drowsiness. A similar machine learning algorithm approach could be used in this case. This would also not require the driver to be wearing any sort of device since the detector could be attached to the vehicle. However, a visual detector may not be as accurate from person to person. Brain waves may be a better indicator for drowsiness since brain activity and alertness have a physiological relationship.
- Another alternative similar to the above approach is to use a motion detector. This detector would record the movements of the vehicle operator to check drowsiness. Again, a machine learning algorithm could be used here as well. One thing that would have to be noted is the movement of the vehicle as well and how it may be affecting the driver and therefore the motion detector.
- One different alternative would be to detect drowsiness based on the operations of the vehicle. Speed, steering, pedal and steering pressures, and lane maintenance can all be recorded and used to check if the driver may be in a drowsy state. With systems like these in place, autonomous vehicle operations such as automatic braking and steering could be deployed to move the vehicle and operator out of potential danger. However, recording this level of data would require many more systems and these systems to be built into the vehicle.

5.4. Impact

Social Impact - The main social impact that will come from the use of the DDDS is safer roads. Since a vehicle operator's level of fatigue can be reliably captured with our system, less accidents would occur due to drivers falling asleep at the wheel. Since our device is

targeted more directly at truck drivers, less accidents would occur involving larger vehicles like semi-trucks and box trucks.

Health and Safety Impact - The EEG design is extremely safe, with only household batteries as a power source. The positive impact on driver safety and accident prevention is the most significant safety impact.

Environmental Impact - The EEG device has little environmental impact, as it only relies on household batteries for power. The software systems have nearly zero environmental impact.

Global and Cultural Impact - A cultural effect is not anticipated, but more adaptive regulation has the potential to make a positive global impact.

Economic concerns - Any drowsy designation after less than 8 hours would cause the driver to be less productive than they originally were, so this is a concern. However, the safety improvement could be profound and would likely lead to a reduction in expensive damages and legal fees.

Ethical Concerns - The main ethical concern with the DDDS is with the data of the drivers and how it is potentially used. Truck driving companies could refrain from keeping drivers who have shown to become drowsy quicker and thus aren't able to drive as much. These companies may even use the data to designate pay changes. Also, the data is sensitive and personal information, so the users of the device should be aware of how it is being used and who or what entities it is being sent to. Users could have the option of opting out of sending their data and only use the data for indicating and alerting of fatigue.

Driver Drowsiness Detection System

Dakota Mouton, Ali Imran, Coady Lewis

FUNCTIONAL SYSTEM REQUIREMENTS

REVISION – I
5 December 2021

Functional System Requirements
FOR
Driver Drowsiness Detection System

Prepared by:

_____
Author Date

Approved by:

_____
Project Leader Date

_____
John Lusher, P.E. Date

_____
T/A Date

Change Record

Rev.	Date	Originator	Approvals	Description
-	10/4/2021	Entire Team		Initial Submission
I	12/5/2021	Entire Team		Original MCU replaced by Raspberry Pi; signal processing sample specs updated

Table of Contents

1. Introduction

1.1. Purpose and Scope

The driver drowsiness detection system is made up of an EEG device paired with an MCU that will gather raw brain activity data and transfer that data to a processing device to validate the proper operation of the device and measure the activity of the brain. It also includes a machine learning algorithm that will be developed and trained to detect various levels of drowsiness from the user, using data gathered from another EEG device that will validate our own design. Figure 1 shows a representative integration of the project in the proposed CONOPS. The verification requirements for the project are contained in a separate Verification and Validation Plan.

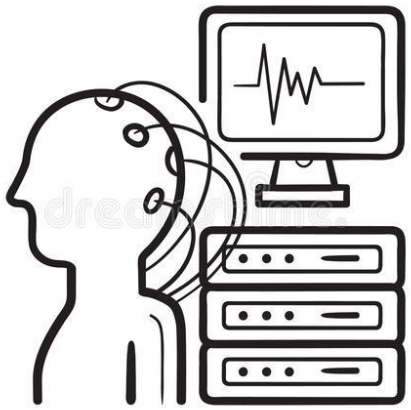


Figure 1. Conceptual Image

The following definitions differentiate between requirements and other statements.

Shall:	This is the only verb used for the binding requirements.
Should/May:	These verbs are used for stating non-mandatory goals.
Will:	This verb is used for stating facts or declaration of purpose.

1.2. Responsibility and Change Authority

Dr. Lusher is our primary authority on changes made to our project and design iterations as he is the project sponsor and lead professor on the project. Each team member is responsible for their unique subsystem outlined in the ConOps report but we work as a team to verify everyone has met the requirements set forth to them.

2. Applicable and Reference Documents

2.1. Applicable Documents

The following documents, of the exact issue and revision shown, form a part of this specification to the extent specified herein:

Document Number	Revision/Release Date	Document Title
1	3.8 10-14-2019	Python Documentation
2	0.4.1	Petal Metrics Documentation
3	2015	IEC 6061 Technical standard for medical electrical equipment
4	2012	IEEE Recommended Practice for Neurofeedback Systems
5	2012	PMC3571819 - Detecting Driver Drowsiness Based on Sensors: A Review

2.2. Reference Documents

The following documents are reference documents utilized in the development of this specification. These documents do not form a part of this specification and are not controlled by their reference herein.

Document Number	Revision/Release Date	Document Title
1	2021.1 07-21-2021	EEGLab Documentation
2	12-09-2019	Muse 2 Manual
3	PIC32MZ	PIC32MZ Family Datasheet

2.3. Order of Precedence

In the event of a conflict between the text of this specification and an applicable document cited herein, the text of this specification takes precedence without any exceptions.

All specifications, standards, exhibits, drawings or other documents that are invoked as “applicable” in this specification are incorporated as cited. All documents that are referred to within an applicable report are considered to be for guidance and information only, except ICDs that have their relevant documents considered to be incorporated as cited.

3. Requirements

This section defines the minimum requirements that the development item(s) must meet. The requirements and constraints that apply to performance, design, interoperability, reliability, etc., of the system, are covered.

3.1. System Definition

Our system is designed to detect a driver’s drowsiness level based on their current brain activity level that is measured through our own EEG device using a machine learning algorithm that we develop. The purpose of this device is to help insurance companies and drivers monitor how awake they really are when driving on long trips across the country for one sitting at a time. Our drowsiness detection system is made up of four subsections: a custom EEG device, a custom microcontroller, simulator and a machine learning algorithm. The EEG device will measure the current brain activity of the user and will comprise four circuits, one for each electrode, that will be attached to the head of the user via a baseball cap or other form of hat. The EEG will send the raw brain activity to the microcontroller which will then send that information to a computer to process the data. The simulator will be made up of a steering wheel, pedals and monitor connected to a computer that will run a version of the game Truck Driving Simulator that we can collect our own data using a commercial EEG device, called a Muse, to build our ML algorithm. The ML algorithm will be trained and built using our collected data that will be tested against datasets from other similar experiments using EEG devices and fatigue detection that could eventually be run on our microcontroller to determine one’s current state without the need of a separate computer.

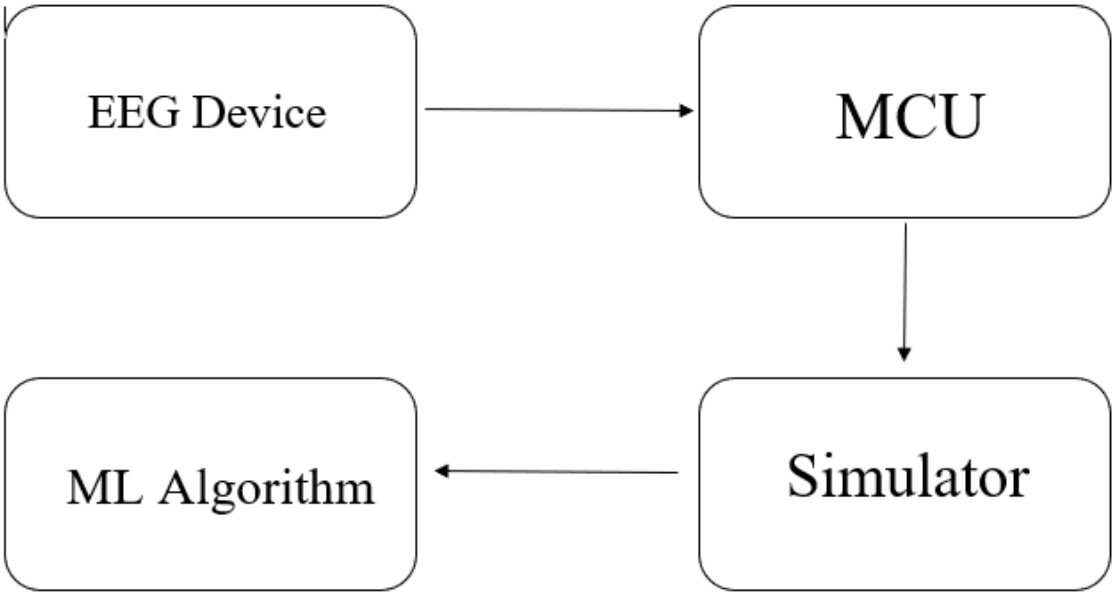


Figure 2. Block Diagram of Driver Drowsiness Detection System

The EEG and MCU will work together to take in the raw brain activity data from the electrodes and transfer that data to a computer for processing and validating that the devices are working as planned. The simulator and ML Algorithm will work together to be developed and trained to detect specific frequencies when a driver might be at one of our three levels of fatigue and give a response based on that level. Once both halves of the project are working on their own, we will then implement them together where the algorithm will be run on our MCU to the determination of fatigue level is all on one system and will not require a separate computer to take in the EEG data and also run the ML algorithm.

3.2. Characteristics

3.2.1. Functional / Performance Requirements

3.2.1.1. Frequency of Measurements

The EEG device will be taking measurements continuously so the algorithm can decide what level of drowsiness the driver is currently at. The Raspberry Pi with external ADC will sample the EEG outputs at a maximum of 5 kHz.

Rationale: We want to measure continuously because we want our algorithm to have detailed features of changes in brain activity because it will result in a more accurate reading of the current state of drowsiness.

3.2.1.2. Accuracy of Measurements

The Drowsiness Detection System will determine the driver's current level of drowsiness with at least 90% accuracy.

> *Rationale: There will be lots of variation in the signals coming from the EEG device so we are taking into consideration those variables we cannot fix or control.*

3.2.1.3. Lifespan and Maintenance

The Drowsiness Detection System will last approximately 8 hours of continuous operation on four 1.5V batteries.

> *Rationale: Similar EEG devices powered by similar battery capacities range from 5 to 10 hours of continuous use time using a rechargeable battery system via usb charging.*

3.2.2. Physical Characteristics

3.2.2.1. Mass

The mass of the Drowsiness Detection System shall be 70 grams or less.

> *Rationale: This is a requirement because of our competing reference Muse device being 61 grams as to not cause strain to the user from wearing the device for prolonged periods of time.*

3.2.2.2. Mounting

The system will be mounted to a baseball cap or similar hat for the user to wear that will have solid contact on each electrode to the skin for accurate measurements.

> *Rationale: As specified by our sponsor, we want the device to be a comfortable wearable device that a driver would not mind wearing for prolonged periods of time while driving.*

3.2.3. Software Characteristics

3.2.3.1. Programming Language

All of the software used in the ML implementation will be written in Python 3.8. Python has many data science, signal processing, and machine learning tools that are free, easily accessible and syntactically simple to use. The Raspberry Pi will be used to implement the machine learning and signal processing python scripts.

3.2.3.2. OS

The entire system will be tested and run using a basic terminal. This increases cross-platform compatibility during design and keeps the implementation lightweight and fast. In testing, we will run the software on a PC. Later, the implementation will be easy to run on a single board computer with any Linux distribution. The Raspberry Pi will use Raspbian.

3.2.3.3. Data Input

During the testing phase, we will collect training data for the ML algorithm with a Muse 2. We will use the Bluemuse and MuseLSL modules to run an LSL stream and get the raw csv values for each channel. We will segment the input into 5 second intervals and perform a short-time Fourier analysis of the signal segments to classify the brain waves.

3.2.3.4. Algorithm Training

We will create our own training data for the algorithm. To do this, we will build a simple driving simulator and monitor each team member with a Muse. Each of us will collect training data in an awake state and separately in a fatigued state.

3.2.3.5. Simulation

We will simulate truck driving with the American Truck Driver software. This is a relatively recent and well-regarded release. We will use a Thrustmaster wheel and pedal set and clamp the wheel to any table so that each team member can perform data collection at home, thus we will not be limited to the FEDC.

3.2.3.6. Algorithm Performance Requirements

Although we will segment the data into 5 second intervals for training with the Muse, for validation, the full process of identifying fatigue will take a maximum of 30 seconds per sample. The data collection step takes a maximum of 10 seconds, the transfer and signal processing step takes a maximum of 10 seconds, and determining the fatigue state will take a maximum of 10 seconds. While the last 2 tasks are sure to take less than 10 seconds each, a buffer is necessary, as the performance costs of going over can be drastic since these processes will be synchronized during integration. Speed increases will have to be tested carefully during integration.

3.2.4. Electrical Characteristics

3.2.4.1. Inputs

On a subsystem level, the EEG will take in voltage signals on the order of microvolts for the brain activity, the Raspberry Pi will take those voltages through an ADC and send them to be processed and the ML algorithm will take in that data and process and determine the current state.

3.2.4.1.1 Power Consumption

a. The maximum peak power of the system shall not exceed 2 watts.

Rationale: This is a requirement specified by our part limits.

3.2.4.1.2 Input Voltage Level

The input voltage level for the Driver Drowsiness Detection System shall be 5V. The input voltage level for the MCU shall be 3V.

Rationale: Enough power to cover the components of our device.

3.2.4.1.3 External Commands

The Driver Drowsiness Detection System shall document all external commands in the appropriate ICD.

Rationale: The ICD will capture all interface details from the low level electrical to the high-level packet format.

3.2.4.2. Outputs

3.2.4.2.1 Data Output

The Driver Drowsiness Detection System shall include a GUI interface for users to observe their current level of drowsiness according to our algorithm.

Rationale: We want to be able to monitor what the levels are looking like and have the customer also be able to see such information about their current drowsiness level.

3.2.4.2.2 Diagnostic Output

The Driver Drowsiness Detection System shall include a diagnostic interface for error detection.

Rationale: Provides the ability to control things for debugging manually.

3.2.4.3. Wiring

The Driver Drowsiness Detection System shall follow the guidelines outlined in IEC 60601-1:2012.

Rationale: Conform to medical device standard.

3.2.5. Environmental Requirements

The Driver Drowsiness Detection System shall be designed to withstand and operate in the environments and laboratory tests specified in the following section.

Rationale: This is a requirement specified by our customer due to constraints of their system in which the Search and Rescue System is integrating.

3.2.5.1. Pressure (Altitude)

The Driver Drowsiness Detection System needs to be able to operate between sea level and 12,000 feet.

Rationale: This is the highest and lowest elevation levels a truck driver might experience in the United States.

3.2.5.2. Thermal

The Driver Drowsiness Detection System can operate between -40°C to +85°C.

Rationale: This is the temperature operation range of our components for the device.

3.2.5.3. Rain

This is a sensitive device and will not be waterproof. Therefore it should not get wet or be submerged in water.

Rationale: The device is not going to be waterproof so it should not take on water.

3.2.5.4. Humidity

The Driver Drowsiness Detection System needs to be able to operate between 0% and 100% humidity.

Rationale: This is the range of humidity a truck driver may experience in the United States.

3.2.6. Failure Propagation

The Driver Drowsiness Detection System will alert the driver when the device is out of battery or no longer working. This will be resolved either by replacing the battery or contacting us for troubleshooting options.

4. Support Requirements

The customer just needs to be able to change out the batteries when the device is dead and a hat to attach the EEG device to.

Appendix A: Acronyms and Abbreviations

BIT	Built-In Test
EEG	Electroencephalogram
GUI	Graphical User Interface
Hz	Hertz
ICD	Interface Control Document
IEC	International Electrotechnical Commission
kHz	Kilohertz (1,000 Hz)
LSL	Lab Streaming Layer
mA	Milliamp
MCU	Microcontroller Unit
MHz	Megahertz (1,000,000 Hz)
ML	Machine learning
mW	Milliwatt
OS	Operating System
PC	Personal Computer
TBD	To Be Determined
V	Volts
W	Watt

Driver Drowsiness Detection System

Dakota Mouton, Ali Imran, Coady Lewis

INTERFACE CONTROL DOCUMENT

REVISION – I
5 December 2021

Interface Control Document
FOR
Driver Drowsiness Detection System

Prepared by:

_____
Author Date

Approved by:

_____
Project Leader Date

_____
John Lusher II, P.E. Date

_____
T/A Date

Change Record

Rev.	Date	Originator	Approvals	Description
-	10/4/2021	Entire Team		Draft Release
I	12/5/2021	Entire Team		Original MCU replaced by Raspberry Pi; EEG powered by four 1.5V AA batteries;

Table of Contents

List of Tables

No table of figures entries found.

List of Figures

No table of figures entries found.

1. Overview

This document describes the interfaces between the different subsystems of the Driver Drowsiness Detection System differentiated between a hardware track and software tack. The hardware track consists of the EEG device and MCU while the software track consists of the ML algorithm. The interfaces inside the HW and SW and between the HW and SW systems will be described here.

2. References and Definitions

2.1. References

- IEC 6061 Technical standard for medical electrical equipment
- IEEE Recommended Practice for Neurofeedback Systems
- PMC3571819 - Detecting Driver Drowsiness Based on Sensors: A Review

2.2. Definitions

V	Volts
mA	Milliamp
Hz	Hertz
kHz	kilohertz (1,000 Hz)
TBD	To Be Determined
C	Celcius
EEG	Electroencephalogram
MCU	Microcontroller Unit
ML	Machine Learning
CSV	Comma-Separated Values
HW	Hardware
SW	Software

3. Physical Interface

Provide details on the physical interface. Examples are:

3.1. Weight

3.1.1 EEG

30 grams

3.1.2 MCU

TBD

3.2. Dimensions

3.2.1 EEG

6.5cm X 5.5cm

3.2.2 MCU

TBD

3.3. Mounting Locations

3.3.1 EEG

The EEG will be a wearable device placed that is attached to a baseball cap or similar hat for the user to wear on their head for the electrodes to maintain contact with their skin.

3.3.2 MCU

The MCU will be a free floating device only restricted by the connections to the EEG and the computer.

4. Thermal Interface

The thermal operating range for the MCU is -40C to 85C. These temperatures will not be reached under its operating conditions and will therefore not require a heat sink.

5. Electrical Interface

5.1. Primary Input Power

5.1.1. EEG

The EEG will be powered by four 1.5V AA batteries. Will be fed into a +-5V voltage regulator and then into the EEG.

5.1.2. MCU

The MCU will be powered by two AA batteries in series. These batteries were chosen due to our need for a 3V and the combination of these batteries in series fulfills this requirement.

5.2. Voltage and Current Values

Component	Voltage	Current
EEG	5V	500 mA
MCU	2.1V-3.6V	200 mA

5.3. MCU

We will be using the PIC32MZ1024EFE100 microcontroller for this project. This device was chosen based on its similarities to the microcontroller recommended by our sponsor. Due to lack of stock, the microcontroller recommended by our sponsor could not be acquired.

5.4. Frequency

The EEG will sample signals ranging from 8 Hz to 32 Hz. The MCU will sample these frequencies at a maximum frequency of 200 MHz

6. Communications / Device Interface Protocols

6.1. Device Peripheral Interface

The EEG will communicate with the MCU via the GPIO pins on the MCU. The MCU will send sampled data to a computer running the AI through a serial port using UART.

7. Machine Learning Interfaces

The following interfaces shall be used together to build and verify the ML algorithm. Out of these interfaces, the Virtual Simulator (7.1), Signal Processing Program (7.4), and ML Algorithm (7.5) shall be part of the end system. Our custom EEG device shall take the place of the EEG Device (7.2) The MCU shall take the place of the EEG Reading Program (7.3) and incorporate the Signal Processing Program and ML Algorithm. The Virtual Simulator will not be part of the end product but shall be used as a means of verification and testing our end product.

7.1. Virtual Simulator

The supervised data used to train and verify the ML algorithm shall come from the use of a truck driving virtual simulator. The virtual simulator will contain the following components assembled onto a rig:

- *Steering Wheel*: controls the steering of the in-game vehicle
- *Pedals*: controls the acceleration and braking of the in-game vehicle
- *Monitor*: displays the first-person point of view inside the in-game vehicle
- *Seat*: holds the driver in place

The steering wheel, pedals, and monitor shall be connected to an external computer which is running an already made truck driving simulation game. The game to be used is American Truck Simulator.

7.2. EEG Device

An already built EEG device shall be used to collect brain wave data from the driver on the virtual simulator. This device shall be connected through bluetooth to an external computer. The EEG device to be used is the Muse 2.

7.3. EEG Reading Program

The computer will receive the brain wave data through a device-reading program. The program shall use the bluetooth connections between the computer and the EEG device to read the data and convert it into a CSV file. The program to be used is BlueMuse.

7.4. Signal Processing Program

The data collected from the virtual simulator shall be filtered to accurately represent and differentiate the different brain wave signals captured from the EEG device. This filtering program shall be written in Python.

7.5. ML Algorithm

The ML algorithm shall be written in Python using the machine learning open-source libraries, TensorFlow. The filtered data collected from the virtual simulator shall be used to train and verify the machine learning algorithm. The specific models used shall be Naive Bayes Classifier, Neural Network, and Kernel SVM.

Driver Drowsiness Detection System Schedule:

Work	End Date	Owner	Status
Create team	8/31/21	All	
Receive project details	9/3/21	All	
Talk with project sponsor	9/9/21	All	
Research	9/14/21	All	
ConOps Report	9/16/21	All	
Assign subsystems	9/21/21	All	
FSR Report	10/4/21	All	
ICD Report	10/4/21	All	
Decide best ML algorithm	10/8/21	Ali, Coady	
Test out Muse 2 device	10/12/21	Ali, Coady	
Create midterm presentation	10/12/21	All	
Get virtual simulator parts ordered	10/13/21	Ali, Coady	
Midterm Presentation	10/13/21	All	
Learn ML algorithm using basic tutorials and online datasets	10/17/21	Ali, Coady	
Create EEG filtering program	10/17/21	Ali, Coady	
Assemble virtual simulator rig	10/19/21	Ali, Coady	
Workout EEG Schematic in Altium for PCB	10/19/21	Dakota	
Finish Circuit Design of EEG	10/19/21	Dakota	
Collect training data from simulator	10/19/21	All	
Test filtering program off of collected data	10/20/21	Ali, Coady	
Get EEG device parts ordered	10/22/21	Dakota	
Have PCB Design Approved and Ready to Print	10/26/21	Dakota	
Create status update presentation	10/30/21	All	

Status Update Presentation	11/1/21	All	
Create ML algorithm off of collected data	11/2/21	Ali, Coady	
Connect EEG to Electrodes and Test	11/2/21	Dakota	
Validate and Troubleshoot EEG	11/2/21	Dakota	
Test ML algorithm with new simulator data	11/7/21	All	
Verify ML algorithm detection rate > 90%	11/14/21	Ali, Coady	
Finished Working EEG	11/23/21	Dakota	
Final validation checks for each subsystem	11/27/21	All	
Create final presentation	1130/21	All	
Final Presentation	12/01/21	All	
Finish final report	12/4/21	All	
Final Report	12/5/21	All	

- Completed
- On Schedule
- Behind

Driver Drowsiness Detection Validation Plan:

Task	Specification	Result	Owner
ML Algorithm drowsiness detection	>90% success rate	85% w/ collected data 97% w/ online data	Ali, Coady
Performance of data collection and processing every interval: ● EEG data collection ● Transfer and signal processing ● Fatigue state output (ML model)	< 30 seconds < 10 seconds < 10 seconds < 10 seconds	Integration Needed <61 ms (0.005 - 0.05 s)	All Dakota Coady Ali
EEG Filters below 8 Hz	f>=8 Hz	See screenshots	Dakota
EEG Filters Above 30 Hz	f<=30 Hz	See screenshots	Dakota
Gain from Instrumentation Amplifier close to 100	G = 80+	G = 1+49,400/Rg, Rg = 560 Ohms, G = 89.2	Dakota
Final voltage readings have amplified from microvolts to volts	0.148<V<0.81172 But V<1 Input 15 Hz, 30uV	Vrms = 0.148 V, Vpk = 0.417 V	Dakota
Final voltage readings have amplified from microvolts to volts (Max and Min Readings)	Max Input: 10 Hz, 30uV Min Input: 1000 Hz, 20 uV	Max: Vrms = 0.594 V, Vpk = 1.67 V Min: Vrms = 4.47 mV, Vpk = 10.5 mV	Dakota
System voltage input	6 V	Regulated to +-5V	All
Peak Power Consumption	2 W	Integration Needed	All

Driver Drowsiness Detection System

Dakota Mouton, Ali Imran, Coady Lewis

SUBSYSTEM REPORT

REVISION – Draft
5 December 2021

Subsystem Report
FOR
Driver Drowsiness Detection System

Prepared by:

_____
Author Date

Approved by:

_____
Project Leader Date

_____
John Lusher II, P.E. Date

_____
T/A Date

Change Record

Rev.	Date	Originator	Approvals	Description
-	12/5/2021	Entire Team		Draft Release

Table of Contents

List of Tables

List of Figures

1. Introduction

The Driver Drowsiness Detection System involves three subsystems, as the MCU subsystem will be replaced by a Raspberry Pi in integration. The project was originally broken into a hardware half (including the EEG device and, formerly, the MCU) and a software half (including the machine learning and signal processing systems). Since each of the three subsystems has been tested and developed as independently as possible, integration should produce a complete and functioning system. Moving forward, and largely before integration, the Raspberry Pi will be handled and programmed as part of the signal processing system.

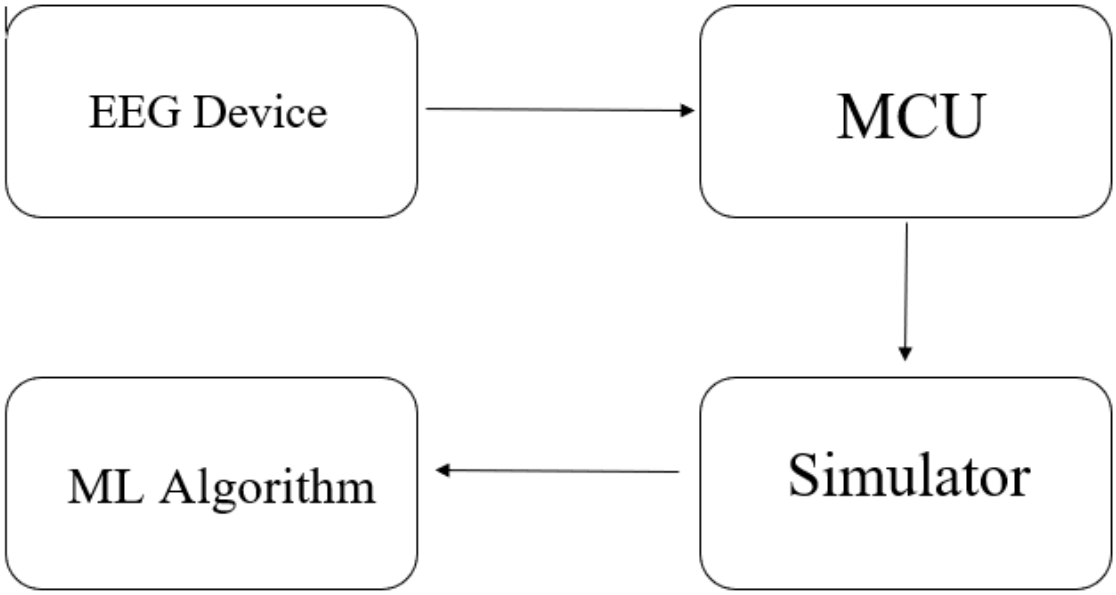


Figure 1: Driver Drowsiness Detection System Block Diagram

2. EEG Device Subsystem

2.1. Introduction

The EEG subsystem consists of a series of op amps,an instrumentation amplifier, and an analog to digital converter to filter out unwanted frequencies from a user's brain activity so we can analyze specific signals for fatigue. The frequencies that we are focusing on are the alpha waves (8-12 Hz) and the beta waves (13-30 Hz) which are both associated with fatigue states. The circuit is powered by four AA 1.5V batteries connected in series to provide a total of 6V to a voltage regulator that will regulate the input voltage to +- 5V to power the components. This also keeps the current of the system low at only around 500 mA from the power supply. The end signal will be connected to the SDA and SCL ports of our MCU for the data to be read and processed.

2.2. Details

2.2.1 The EEG Circuit

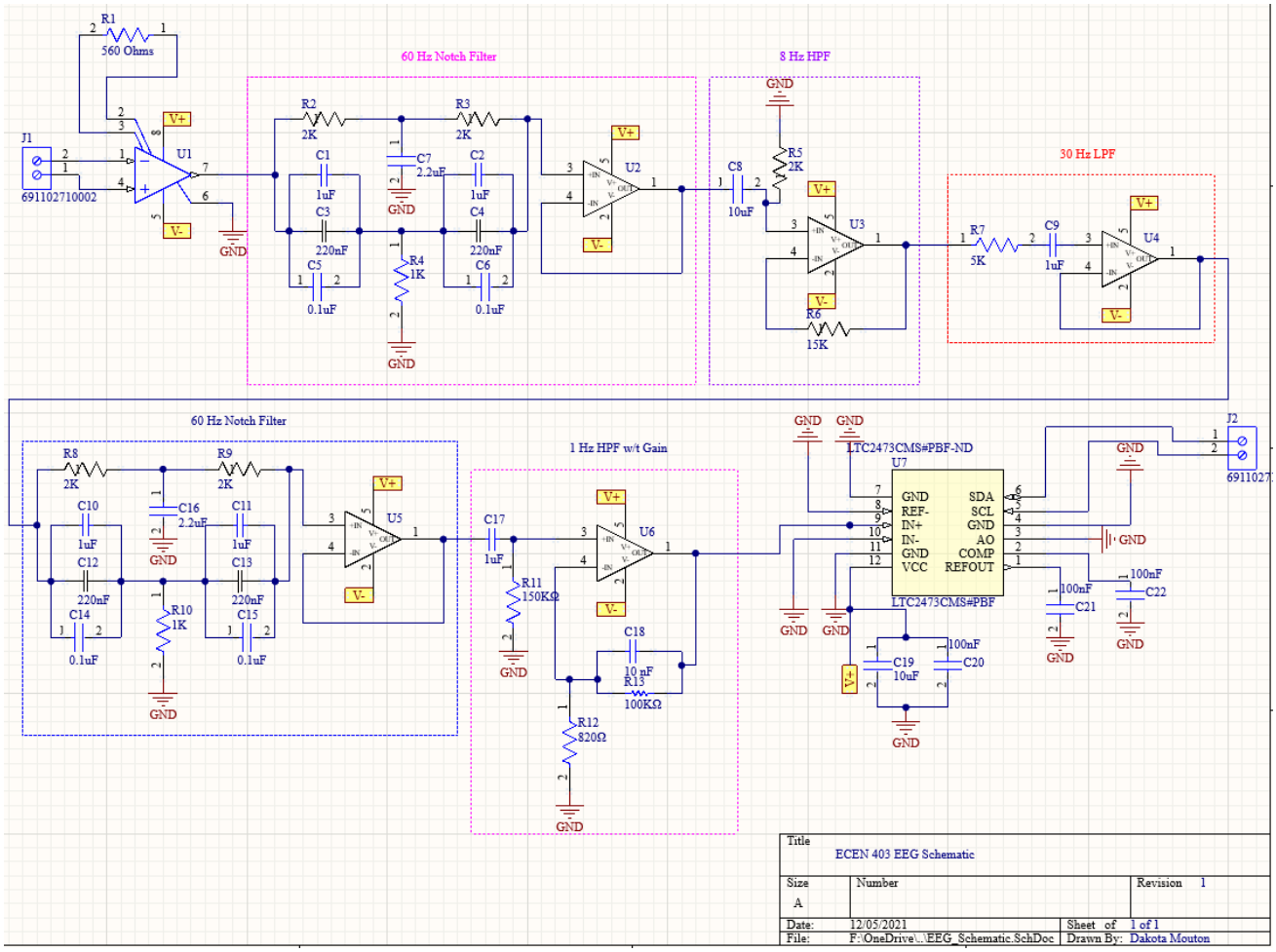


Figure 2: Finished EEG Circuit

The circuit of the EEG begins with an instrumentation amplifier that takes in the electrode signals as inputs and applies 89.2X gain to the signal using a 560 Ohm resistor. The gain equation for our circuit is 1+49,400/Rg ,where Rg is the gain resistor, which results in our target gain of around 90. We apply this initial amplification because the input signal is in the range of microvolts and will be hard to process the signal accurately at that range. This amplification gets our signal into the millivolt range which is more usable.

Figure 3: 60 Hz Notch Filter Bode Plots

The next section of the circuit is our first 60 Hz Notch filter which is designed to filter out specifically the 60 Hz frequency from our input signal. We have two of these filters in our circuit for potential powerline interference from our power source. We want as clean a signal as we can get and this helps remove some of the unwanted noise.

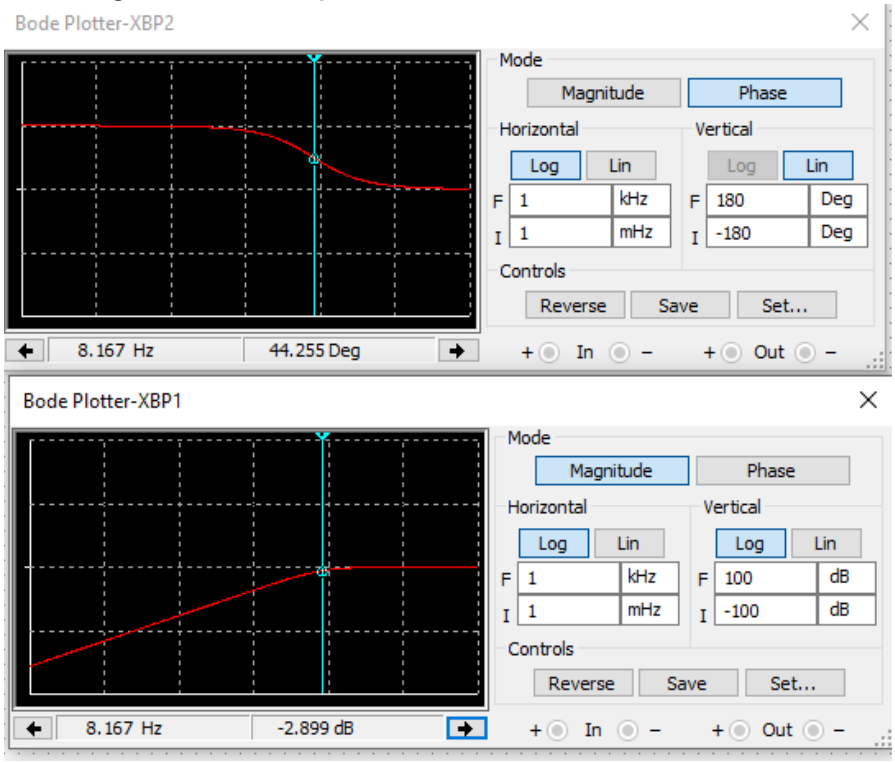


Figure 4: 8 Hz High Pass Filter Bode Plots

The high pass filter is designed to filter out frequencies below the 8 Hz threshold. When making a high pass filter we want our phase angle to be around 45 degrees and our magnitude to be around the 3 dB range for an optimal filter. We can see from the simulations that our cutoff frequency is around 8.167 Hz which is right on target.

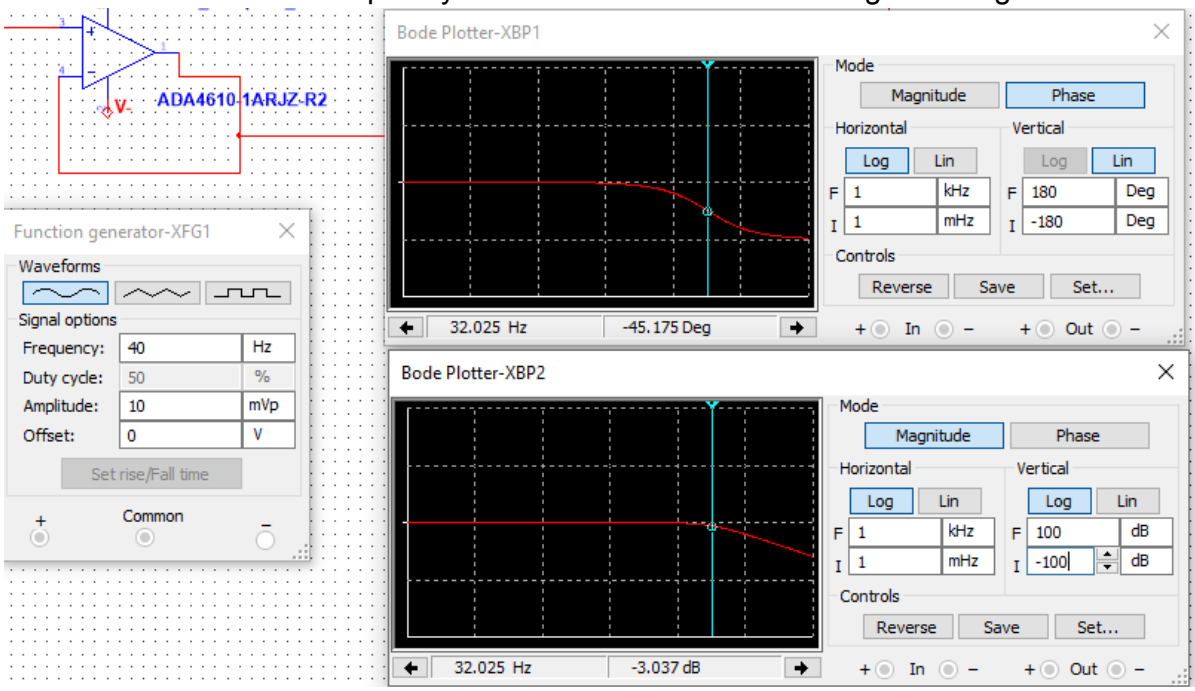


Figure 5: 30 Hz Low Pass Filter Bode Plots

The low pass filter removes the frequency signals above the 30 Hz range. We optimally want to reach our result around similar conditions to the high pass filter with the phase angle of around 45 degrees and the magnitude of 3 dB. We ended up with a cutoff frequency of 32.025 Hz from our simulation data.

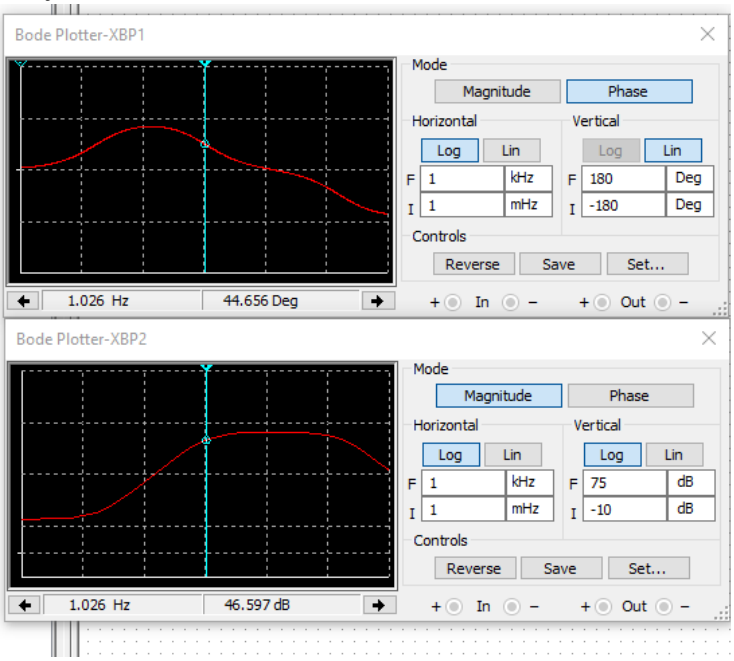


Figure 6: 1 Hz High Pass Filter Bode Plots

The 1 Hz high pass filter is used to reduce some extra noise that could potentially be found from the input signals, as they are very sporadic. This component also adds the final amount of gain to our circuit to reach our target voltage level in the millivolt range. There is going to be some voltage across some of these components so some final gain at the end of the circuit helps amplify the signal to a more viewable and workable output.

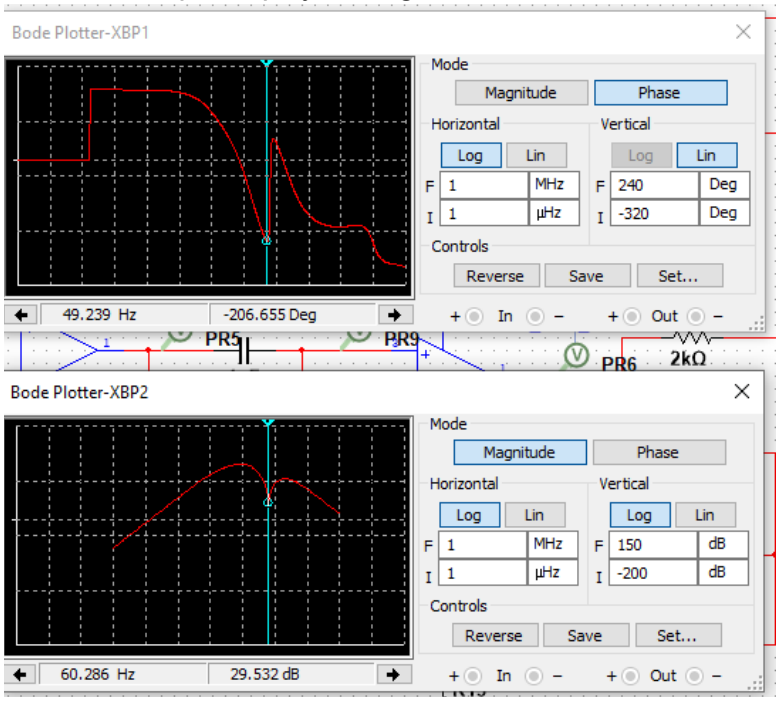


Figure 7: Full EEG Output

Based on the phase bode plot we can see the flat line on the graph as the signals we are focusing on (8Hz to 30 Hz) with a dip in the graph around 60 Hz for the rest of the filtering.

2.2.2 The PCB Design

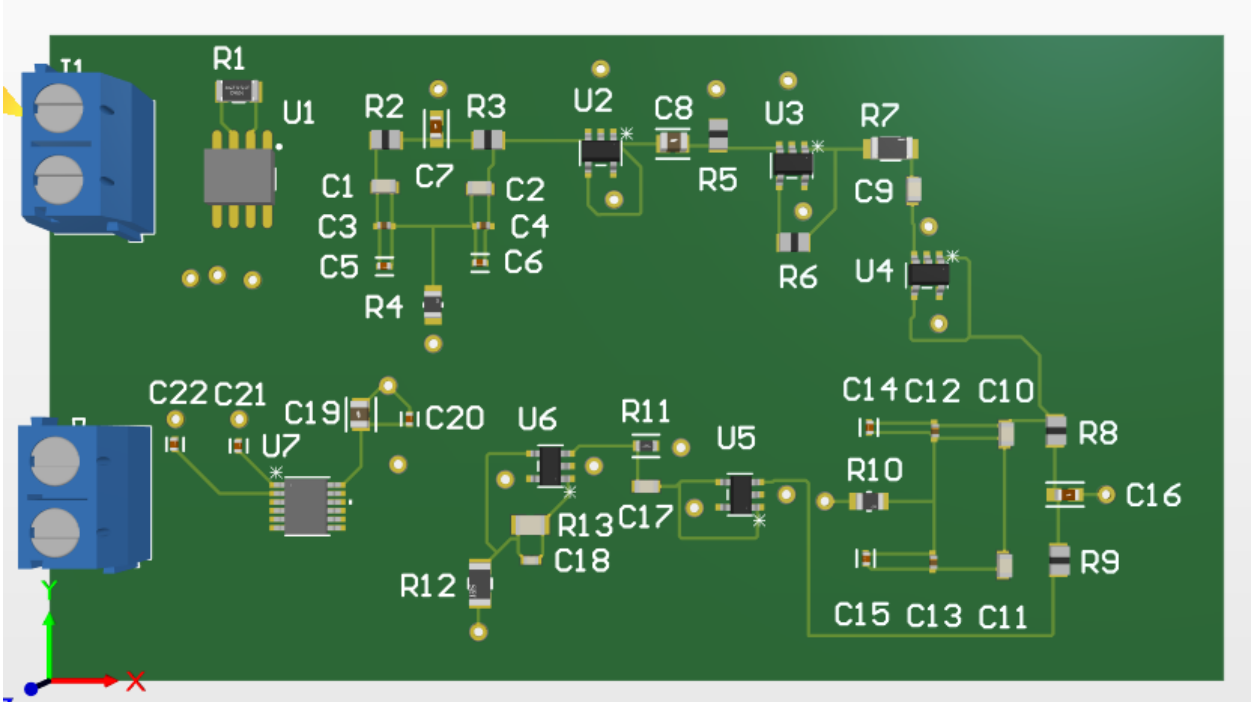


Figure 8: PCB Design

The PCB design for this circuit is still a work in progress and will be finished for 404 when we have to implement our subsystems. The PCB has had to go through a few redesigns due to errors from simulations from earlier in the semester and issues with the breadboard circuit as well. Now the simulations are working completely as they should and have figured out what components we need to finish the new circuit design, the PCB should not be too far behind to follow and get ordered.

2.3. Validation

2.3.1 Simulation Results

8Hz_HPF

Input Frequency : (Hz)	Input Voltage : (mV)	RMS Voltage : (mV)	Cut-off Frequency: (Hz)	Phase shift: (Deg)	Magnitude: (dB)	Output Voltage: (mV, RMS)
5	5.85	4.14	8.591	45.823	-3.127	2.04
5	10	7.07	8.591	45.823	-3.137	3.49
5	20	14.1	8.591	45.823	-3.127	6.98
10	5.85	4.14	8.167	47.271	-3.369	3.1

15	**5.85**	**4.14**	**8.591**	**45.823**	**-3.137**	**3.57**
15	**10**	**7.07**	**8.591**	**45.823**	**-3.137**	**6.1**
15	**20**	**14.1**	**8.591**	**45.823**	**-3.137**	**12.2**

30Hz_LPF

Input Frequency : (Hz)	Input Voltage : (mV)	RMS Voltage : (mV)	Cut-off Frequency: (Hz)	Phase shift: (Deg)	Magnitude: (dB)	Output Voltage: (mV, RMS)
10	**5.85**	**4.14**	**32.025**	**-45.175**	**-3.037**	**3.95**
10	**10**	**7.07**	**32.025**	**-45.175**	**-3.037**	**6.74**
10	**20**	**14.1**	**32.025**	**-45.175**	**-3.037**	**13.5**
40	**5.85**	**4.14**	**32.025**	**-45.175**	**-3.037**	**2.57**
40	**10**	**7.07**	**32.025**	**-45.175**	**-3.037**	**4.39**
40	**20**	**14.1**	**32.025**	**-45.175**	**-3.037**	**8.79**

60Hz_Notch Filters

Input Frequency : (Hz)	Input Voltage : (mV)	RMS Voltage : (mV)	Cut-off Frequency: (Hz)	Phase shift: (Deg)	Magnitude: (dB)	Output Voltage: (mV, RMS)
60	**5.85**	**4.14**	**60.06**	**-19.319**	**-25.02**	**0.136**
60	**5.85**	**4.14**	**63.096**	**-3.486**	**56**	**0.136**
30	**5.85**	**4.14**	**60.06**	**-19.31**	**-25.02**	**1.57**

				9		
30	5.85	4.14	60.06	-3.486	56	1.57
59	5.85	4.14	60.06	-19.319	-25.05	0.161

1Hz_HPF

Input Frequency: (Hz)	Input Voltage: (mV)	RMS Voltage: (mV)	Cut-off Frequency: (Hz)	Phase shift: (Deg)	Magnitude: (dB)	Output Voltage: (mV, RMS)
1	5.85	4.14	0.975	45.378	46.467	863
1	10	7.07	1.026	44.656	13.96	1180
10	5.85	4.14	0.975	45.378	46.467	1200
10	10	7.07	1.026	44.656	13.69	1270

Full EEG Circuit

Input Frequency: (Hz)	Input Voltage: (mV)	RMS Voltage: (mV)	Cut-off Frequency: (Hz)	Phase shift: (Deg)	Magnitude: (dB)	Output Voltage: (mV, RMS)
10	0.02	0.0141	54.483	-203.193	29.532	396
30	0.02	0.0141	54.483	-203.193	29.532	99.3
75	0.02	0.0141	60.286	-155.308	29.532	9.35
100	0.02	0.0141	60.286	-155.308	29.532	15.4
300	0.02	0.0141	60.286	-155.308	28.248	23.1

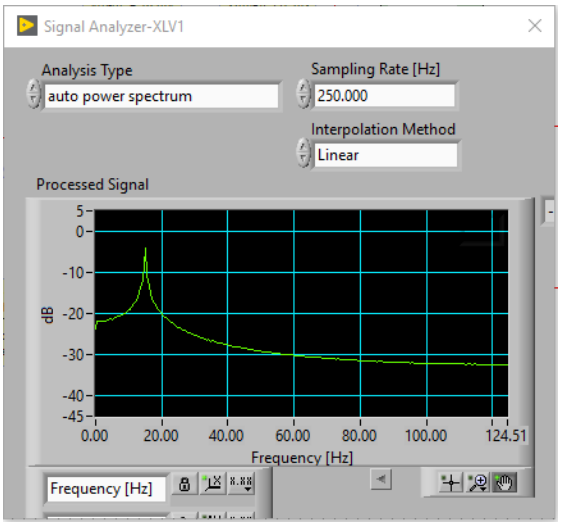
1000	**0.02**	**0.0141**	**60.286**	**-155.308**	**29.532**	**4.47**
10	**0.03**	**0.0212**	**60.286**	**-155.308**	**29.532**	**594**
100	**0.03**	**0.0212**	**60.286**	**-155.308**	**29.532**	**19.1**

Full EEG Circuit (Switched)

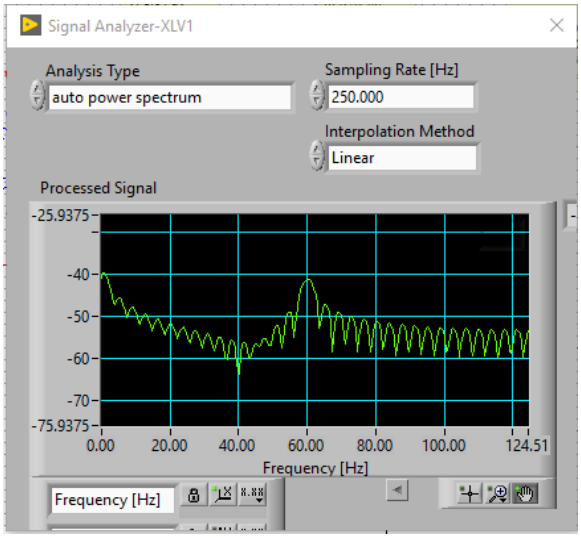
Input Frequency: (Hz)	Input Voltage: (mV)	RMS Voltage: (mV)	Cut-off Frequency: (Hz)	Phase shift: (Deg)	Magnitude: (dB)	Output Voltage: (mV, RMS)
15	**0.03**	**0.0212**	**60.286**	**-155.042**	**19.682**	**148**

2.3.2 Simulation Graphs

a.)



b.)



c.)

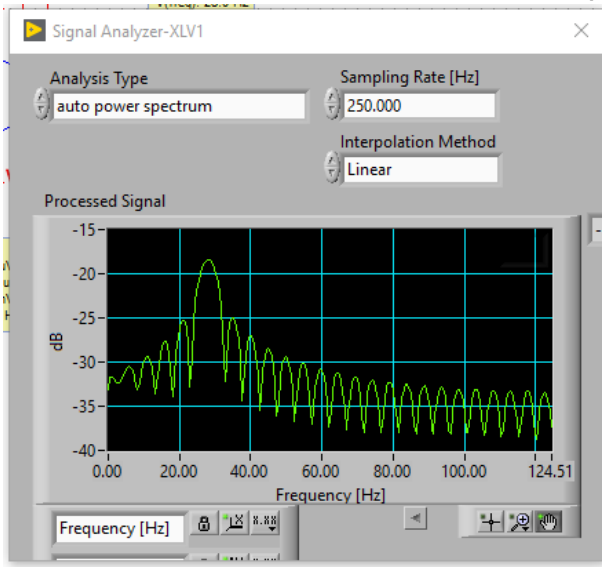


Figure 9: a.) 15 Hz test input, b.) 60 Hz test input, c,) 30 Hz test input

When testing the circuit via simulations I could only input one test frequency at a time. When looking at these graphs here with the various test input frequencies, we can see the spikes around those frequencies and more filtering around the other ones, showing a focus on our target frequency. Which means when applied to the physical board we should see similar results.

2.3.3 Breadboard Tests

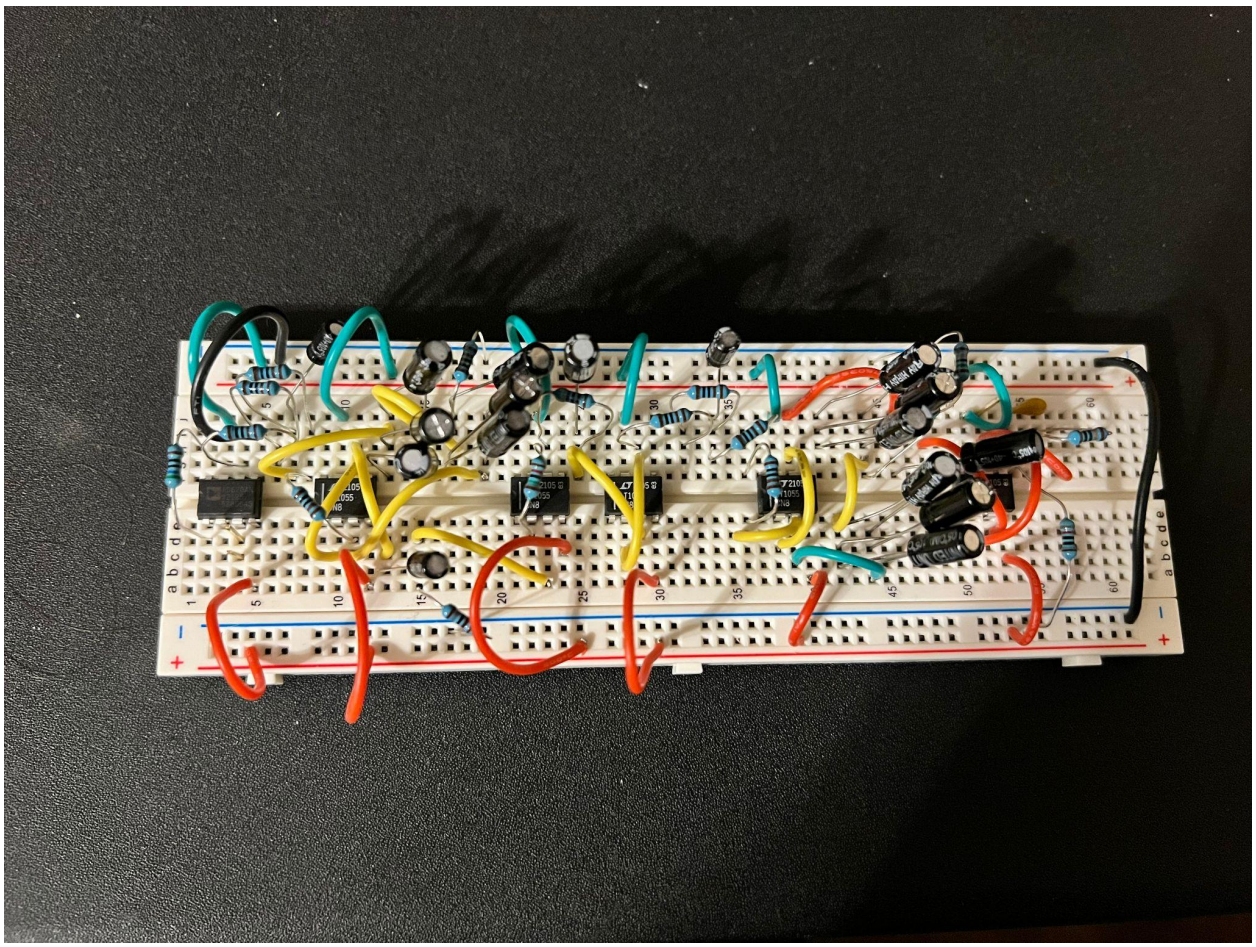


Figure 10: Breadboard of EEG Circuit

Unfortunately I have not been able to properly test if the breadboarded version of the circuit works correctly. During our demo I was able to show target frequencies on the graph similar to Figure 7, however I was not able to properly demonstrate the live frequencies using the electrodes due to not being able to get them to work. Without the electrodes working I could only test one frequency at a time just like the simulations so I could not give an accurate demonstration of the circuit. I believe the issue with the electrodes was not being able to get solid contact with the skin, due to me having lots of hair in the optimal locations on the back of the head, or spots O1 and O2 from Figure 1 in the Signal Processing Subsystem section. Future improvements will be made to making the electrodes to have them come in contact with a more open part of the head for a better reading.

2.4. Conclusion

As of now the EEG subsystem has a completely working simulated circuit with various tests run on it giving acceptable results in the end. Various test input frequencies and voltages were given to make sure the circuit handled any possible situation that it would run into for our application. The breadboard still needs some work with getting the electrodes working for a live feed of brain activity which will be the focus over winter break to get working. Once that has been done completely then the PCB design can be finished and sent off to be printed and solder our components on to get our final product for

implementation. A current limitation of this circuit's design is that it only uses two electrodes for input so it cannot grab the most accurate data. So we will be putting together another one or two boards to get us to four or six total electrodes for more accurate measurements.

3. Machine Learning Model Subsystem

3.1. Introduction

The purpose of the Machine Learning Model Subsystem is to take in processed EEG signals and output one of two classifications of a user's mental state: awake or drowsy. Three different machine learning models were created, trained, and tested with various datasets. These models are the Naive Bayes classifier, Artificial Neural Network, and Kernel Support Vector Machine.

3.2. Details

3.2.1. Data Collection

Data was collected through the use of a truck driving simulator. The simulator included a set of pedals and steering wheel hooked up to a computer. The computer was running the game, *American Truck Simulator*, to simulate the first-person experience inside a truck. During data collection, the subjects wore the Muse 2 EEG device, and the device was connected to another computer which read in the EEG samples into comma-separated value (.csv) files.

Figure 11: Simulator setups with subject 1 (Ali I.) on the left and subject 2 (Coady L.) on the right

One dataset contained the data of one subject (Ali I.). The dataset contained roughly 4 hours of awake data and 4 hours of drowsy data. Most of the data came from 1-hour intervals where the subject wore the Muse device and collected simulated driving data in hour long sessions. These sessions started when the user felt they were awake and when the user felt they were drowsy.

Another dataset contained the data of two subjects (Ali I. and Coady L.). The dataset contained roughly 2 hours of awake data and 2 hours of drowsy data - 1 hour of awake and 1 hour of drowsy from each subject. The data came from 15-minute intervals as opposed to

the 1-hour intervals from the previous dataset. This method allowed the data to be better classified since shorter intervals are much easier to correctly define as either awake or drowsy data. It also prevented large volumes of data from being thrown out due to possible errors with how the Muse was placed.

Another dataset was obtained from an online dataset containing raw signals of 14 different people: 7 male and 7 female. Each person had a 10 minute sample of awake data and a 10 minute sample of drowsy data for a total of 140 minutes of awake and 140 minutes of drowsy data. Since this dataset is not derived from simulated driving, it does not completely represent the circumstances that the model shall end up being used in.

3.2.2. Data Preprocessing

The raw EEG signals from the datasets have been processed by the Signal Processor. The collected data is in the form of 12 features: 3 features (alpha/beta, beta/theta, alpha/theta ratios) for each of the 4 electrodes from the Muse. The online data is in the form of 40 features: 1 feature (alpha/beta ratio) for each of the 40 electrodes. Before inputting the data into the ML models, the data first needed to be preprocessed.

After the online dataset was processed by the Signal Processor, many feature columns contained null values. After removing all columns that contained any null values, the number of features dropped from 40 to 23.

The first main step in the preprocessing was pruning the data. One issue with the data is that many of the features are right skewed with a long tail towards the maximum value. Each of the models relies on the separation and width between values, and having such a large tail causes a lot of the data to be less distinct between the boundaries. To counteract this, entries with feature values outside of 3 degrees of standard deviation were removed.

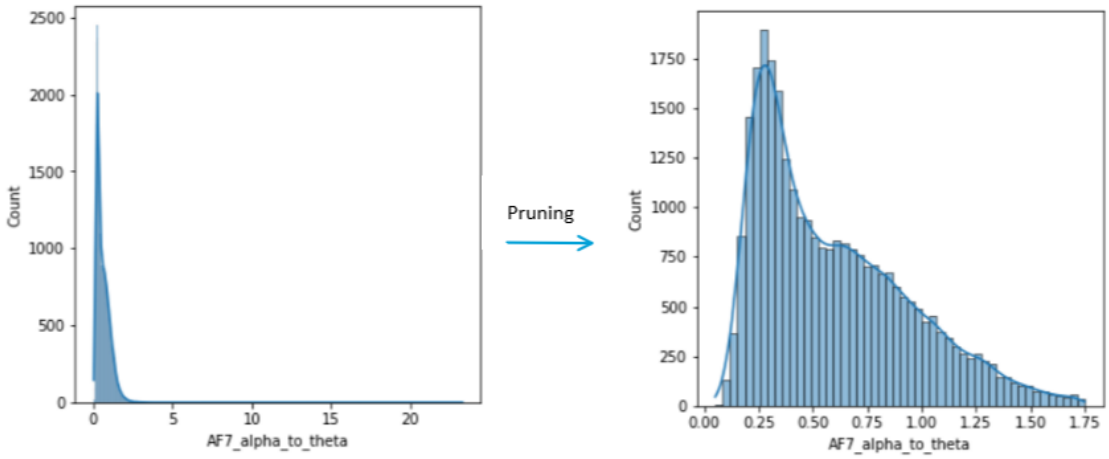


Figure 12: Histograms of a feature (electrode AF7 alpha/theta values) before and after pruning

Before inputting the data into the Neural Network and Kernel SVM, the data needs to be scaled down. Both models work better when each feature is normalized to be within the set [0, 1]. This is done by taking the minimum and maximum values in each feature and normalizing each data point with those values.

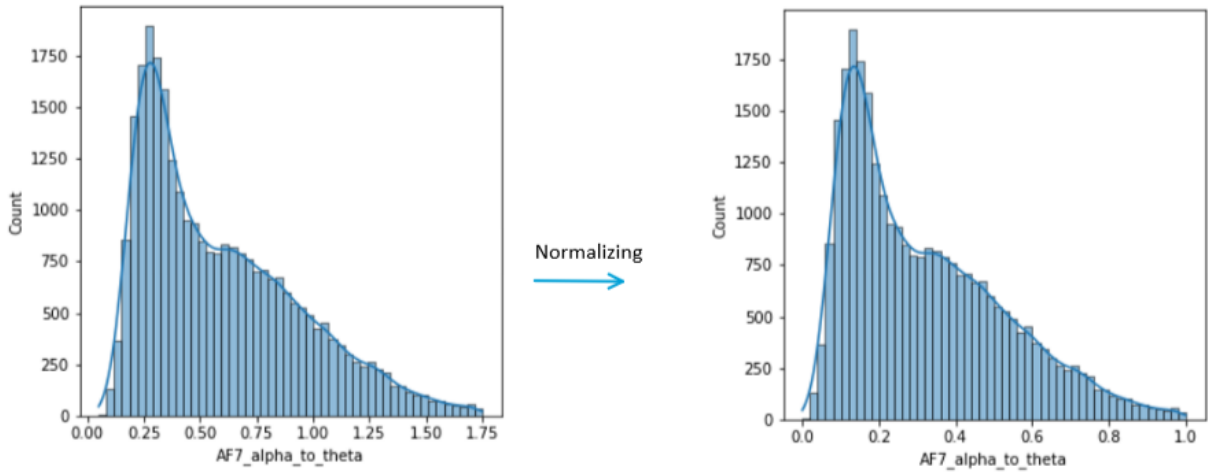


Figure 13: Histograms of a feature (electrode AF7 alpha/theta values) before and after normalization

3.2.3. ML Models

3.2.3.1. Naive Bayes Classifier

The Naive Bayes classifier is built off of the Naive Bayes' Theorem. The theorem is based on prior probabilities, posterior probabilities, and likelihood of a feature's presence and its relation to the outcome.

3.2.3.2. Artificial Neural Network

An artificial neural network is a machine learning model which contains a network of nodes across multiple layers. Each node has its own weights and biases, and the activation of a node designates where data is sent to next in the network.

After experimenting with multiple different parameters, the network's final architecture ended up with 4 layers: 1 input layer, 2 hidden layers, and 1 output layer. The network was built in an upside-down pyramid fashion where less nodes were introduced as the layers deepened. The output layer is a single node which holds a value between 0 and 1 where an output < 0.5 is classified as 0 or "awake" and an output > 0.5 is classified as 1 or "drowsy". The number of nodes in the input and hidden layers were experimented with. The final number of nodes were based on the number of features of the datasets:

- Input: # features + 1
- Hidden layer 1: (# features / 2) + 1

- Hidden layer 2: (# features / 4) + 1

Basing the number of nodes on the number of features allows for both the Muse collected dataset and online dataset to use the same network structure.

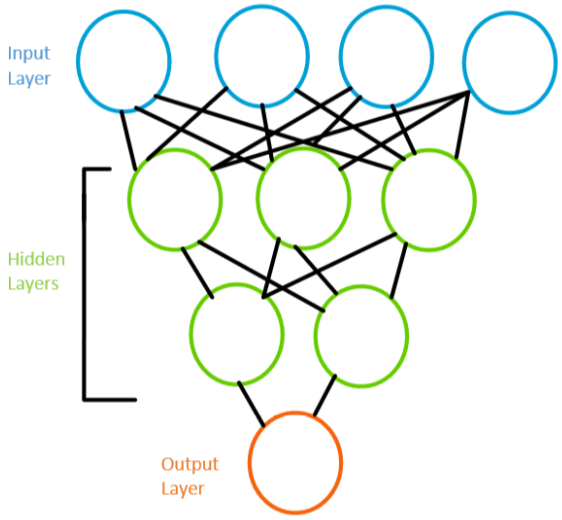


Figure 14: Basic representation of the Neural Network's architecture

As the network is training, it is validated across 20% of the entire dataset.This validation data helps to tune the network's nodes' weights across new data which helps the network to generalize across data better. Another method that was introduced to reduce overfitting was the addition of dropout layers. These layers were placed directly after the hidden layers. While training in every iteration, nodes are randomly "dropped" in the hidden layers to reduce biasing across certain nodes in the network. A dropping rate of 20% was chosen after experimenting with multiple values.

3.2.3.3 Kernel SVM

The Kernel Support Vector Machine is a classifier which takes in a non-linear feature set and transforms the set into a higher dimensional space with a kernel where a non-linear boundary between classes can be constructed. After experimenting with different kernel types, the chosen kernel type for the final model was the Radial Basis function.

Validation of the training and testing capabilities of the Kernel SVM came in the form of k-fold cross validation. K-fold cross validation involves splitting up the training data into K folds. The model is then trained on K-1 folds and uses the last fold to test on. This validation is done K times where each fold ends up as a test fold. This process verifies the generalization ability of the model on unseen data. A relatively low standard deviation across the K test accuracies indicates that the model is not overfitting or relying purely on trained values. The Kernel SVM model was validated across 10 folds.

3.3. Validation

The Machine Learning models were validated on two main metrics: accuracy of the model on unseen data and speed of prediction.

3.3.1. Accuracy

The target accuracy of creating a model that could correctly output an EEG signal as either awake or drowsy was 90%. The models were tested on new, unseen data. This was done by randomizing and splitting the data into a training set and testing set. For the Naive Bayes and Kernel SVM classifiers, 80% of the data was used to train the model, and the other 20% was used to test the model. For the Neural Network, 60% of the data was used to train the model, 20% was used to validate the model as it was training, and the last 20% was used to test the model.

Accuracies (%)	Dataset		
Model	8 Hours Ali	4 Hours Ali/Coady	Online Dataset
Naive Bayes	71.7	74.6	72.9
Neural Network	78.3	83.5	97.1
Kernel SVM	**83.5**	**85**	**97.3**

Table 1: Testing accuracies of the ML models over various datasets

For all models, the accuracy over the 4 hour combined-subject dataset was greater than the accuracy over the 8 hour single-subject dataset. This seems unexpected at first given that more data over a single subject should be more accurate than less data over multiple subjects. However, the 4 hour combined dataset is built off of data collected over 15-minute intervals rather than 1-hour intervals in the 8 hour single dataset. This comparison verifies that classifying data as awake and drowsy in 15-minute intervals is better than the 1-hour intervals which are prone to errors in initial classification.

For all datasets, the Kernel SVM ended up most accurate. The Kernel SVM achieved an 85% accuracy from data collected off Muse and an accuracy of 97% with the online dataset. Although the Neural Network performed slightly worse than the Kernel SVM, one advantage it has is the number of configurable hyperparameters. Even though these hyperparameters add complexity to the model, they also allow the model to be further tuned to the point where it may end up becoming more accurate than the Kernel SVM.

The online dataset ended up with the highest accuracy. The target accuracy of 90% was achieved using the online dataset but just out of reach with the Muse data. Since the

online dataset is collected from a 40-electrode EEG device, more features can be extracted than from the 4-electrode Muse EEG device. Both EEG devices cannot be directly compared electrode-wise since the Muse contains electrodes over the forehead and behind the ears whereas the online dataset EEG device contains electrodes over the scalp and around the head.

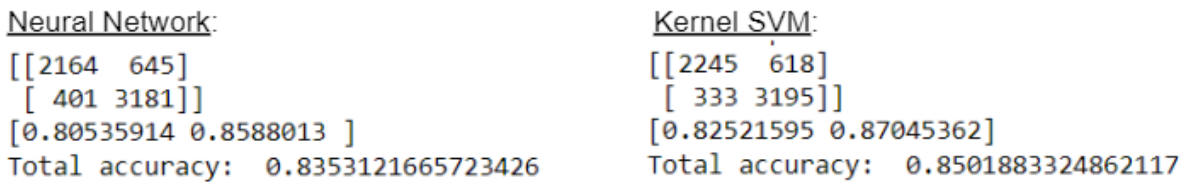


Figure 15: Confusion matrices of Neural Network and Kernel SVM over the combined (subject 1 + 2) dataset

The confusion matrices above showcase the number of true negatives (top left), false positives (top right), false negatives (bottom left), and true positives (bottom right) for the Neural Network and Kernel SVM over the combined dataset. For both models, the accuracy of detecting drowsy (positive) was higher than detecting awake (negative). A higher rate of false positives is much better than a higher rate of false negatives. False positives are moments where a user is awake but is alerted as drowsy. Ideally, the models should have a low false negative rate where a user is drowsy but is labeled as awake.

3.3.2. Speed

Speed was used as another metric for validation because classification of EEG signals will eventually happen in real-time. Therefore, the model needs to be able to predict a set of data within a certain time frame that does not surpass the time frame of the data collected and the time it takes for the EEG to send the data to the Signal Processor. The target prediction speed was established as predicting in <10 seconds over a 30 second chunk of data. Since the Naive Bayes algorithm's accuracies were considered low, the speed of just the Neural Network and Kernel SVM was looked at. Testing the prediction speed was done by taking a random 30-second chunk of processed EEG data and timing how long predicting values over that data took.

Model	Speed (s)
Neural Network	0.005
Kernel SVM	0.05

Table 2: Prediction speed of the ML models over 30 seconds of data

Both the Neural Network and Kernel SVM performed well under the limit of 10 seconds for predicting a 30-second chunk of data with the network at 5 ms and the SVM at 50 ms. The Neural Network was roughly 10 times faster than the SVM.

3.4. Conclusion

After experimenting with multiple machine learning models, the Kernel SVM ended up being the most accurate across testing data. Although the Kernel SVM is slower than the Neural Network, its prediction time is still well under the specified limit. As mentioned above, the advantage of using a Neural Network is in tuning its hyperparameters to possibly gain more accuracy. Moving forward into integration, both the Kernel SVM and Neural Network will be combined with the Signal Processor to more efficiently tune the parameters within the Signal Processor and observe the effects (increase or decrease in accuracy) over each of the models.

4. Signal Processing Subsystem

4.1. Introduction

The signal processing subsystem is vital for extracting meaningful features from raw EEG signals. This is because fatigue manifests in the signal as changes in activity of established frequency bands, and the raw signal is nearly incomprehensible (figure 1). The bands pertinent to fatigue detection include: theta (4-8 Hz), alpha (8-12 Hz), and beta (12-30 Hz). In general, beta corresponds to more focused activities, alpha to a state of relaxation and drowsiness, and theta to a state of extreme relaxation and sleep. These biological signals are not always predictable, thus the need for a machine learning classifier. In addition to feature extraction, the final iteration of our project requires a real time analysis of a signal stream to classify the user's current fatigue state, so this subsystem implements that time-frequency analysis. For development of the subsystem, a Muse 2 EEG band was used to collect raw signals. The live analysis was tested for extended continuous operation, accuracy vs the recording analysis, and robustness to a failure of the data feed. The recording analysis was also tested to ensure that it isolated the frequency bands properly with a common alpha-band response test.

4.2. Details

Beginning with the recording analysis, as the live analysis was an extension of this, BlueMuse and MuseLSL were used to connect to the Muse via bluetooth and save recordings to a csv file. The Muse sampled at 256 Hz, and each recording included the voltage values in mV at each of four electrodes, TP9 and TP10 behind the ears along with AF7 and AF8 on the forehead. The reference electrode for all of these measurements was a fifth electrode in the center of the forehead. All of this was constrained by the design of the Muse.

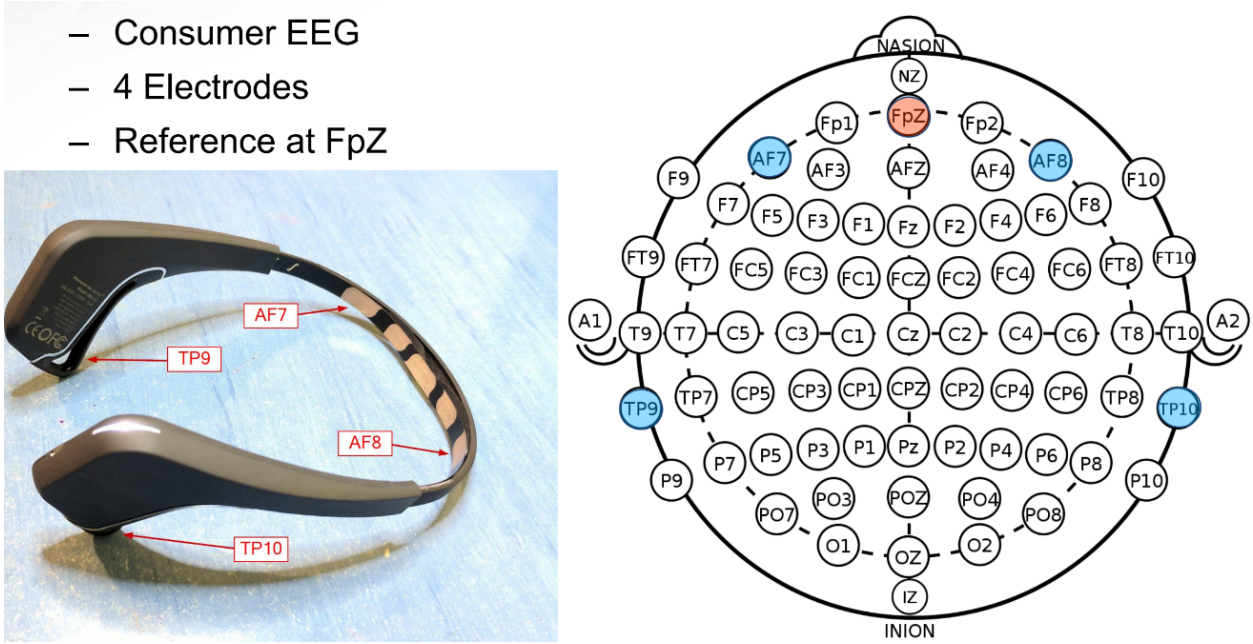


Figure 16: Muse Electrode Placement on International 10-40 Standard

A function was written to place the data from a CSV recording in a 2d list for processing, and another to write back to a new file in the same format as the original CSV file. Then, another function was written to compute the fast fourier transform magnitude of a subset of the signal. This function returned a discrete FFT magnitude plot in the form of a list of positive frequency values and the FFT magnitude at each of those frequencies. Another function was used to calculate the bandlimited average power over the domain of each subset (i.e. if the FFT came from a 5s subset of the signal, the average taken was of the frequency band's power in that subset, not over the entire signal interval). The last helper function calculated the ratio of power between two frequency bands given a data array, electrode of choice, index interval of the signal subset, and the two frequency bands of interest.

For the feature output, the window size and increment parameters were fixed and three power ratios were returned from each window, the alpha/beta, theta/beta, and alpha/theta ratios. As an example, with a window size of 1280 samples and an increment parameter of 256 (5s and 1s respectively at 256 Hz), the output includes the three power ratios in the following subsets of the signal: the 1st-1280th samples, 257th-1536th samples, 513th-1792nd samples, and so on until the end of the signal recording. The overlap was introduced to get more data about how the bandpower ratios propagated in time while keeping the stability of a larger window size. This is the fundamental difference between the recording analysis and live analysis programs. The former's purpose is to extract as much data as possible to train the ML model, while the latter's purpose is to use an already trained model to classify the current fatigue state, thus the live analysis omits overlap by setting the window parameter equal to the increment parameter.

A wavelet transform implementation was attempted early on, but the accuracy of the ML classifier was much lower than the current short-time Fourier transform with overlapped rectangular windows (one test as low as 67% compared to >80% for the STFT), so the wavelet transform was scrapped.

The first iteration of the recording analysis calculated the alpha/beta ratio and second calculated the individual theta, alpha, and beta power values. The final iteration has a mode selection parameter to implement any of the three calculations, and it can also handle any number of electrodes, such that online datasets could be used for training.

The live analysis code implements these same functions, but on an updating CSV file rather than a completed recording. To do this, an additional function was added to skip to a line of interest in a CSV file in a very short amount of time. This function returns the data from that line of interest to the end of the CSV file, along with the last line read. From this, live analysis could pull any updates from the data file and would do so every time the file's last modified time was changed. In order to eliminate synchronization issues, the code depends on the file being updated at consistent intervals. This is how the Muse writes data, and it will be simple to set the timing of the updates with the microcontroller next semester. I added a check for synchronization in the code, but as a few samples out of over 1000 will not significantly change the output, there is tolerance built in, set by default to 10 samples.

4.3. Validation

To check that the bandlimited power calculation was properly extracting the frequency bands, a common test on the alpha band was performed. It is known from established medical research that the alpha power spikes when the subject closes their eyes, and returns to normal when they open them. This is because cutting off visual input causes immediate relaxation in the occipital cortex, producing strong alpha waves. The figure below considers only TP9 (orange) and TP10 (blue), as those electrodes are closest to the occipital cortex.

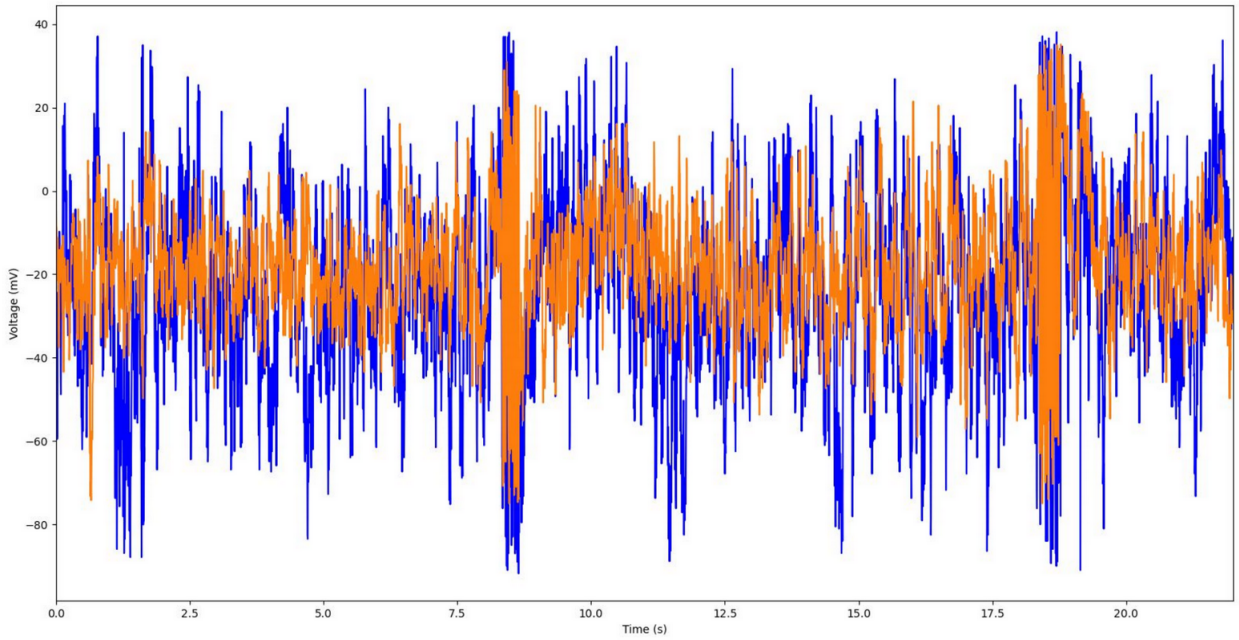


Figure 17: Raw EEG Signal from Eye Test

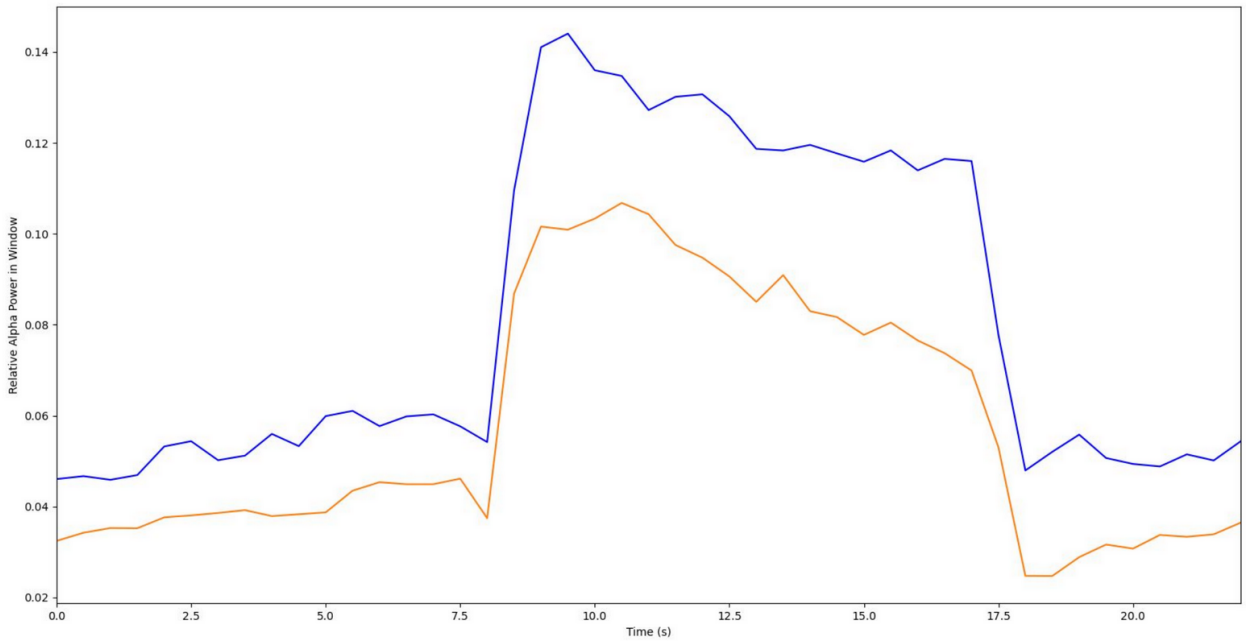


Figure 18: alpha/beta Power Ratio from Eye Test

Note that there are magnitude spikes in the raw signal when the subject's eyes are closed at 8s and opened at 17s. This is due to the action of opening and closing the eyes causing extra voltage on the surface of the skin, but there is otherwise no indication of what action took place, as any muscle movement could have caused those spikes. The alpha/beta ratio provides a more useful description of the signal.

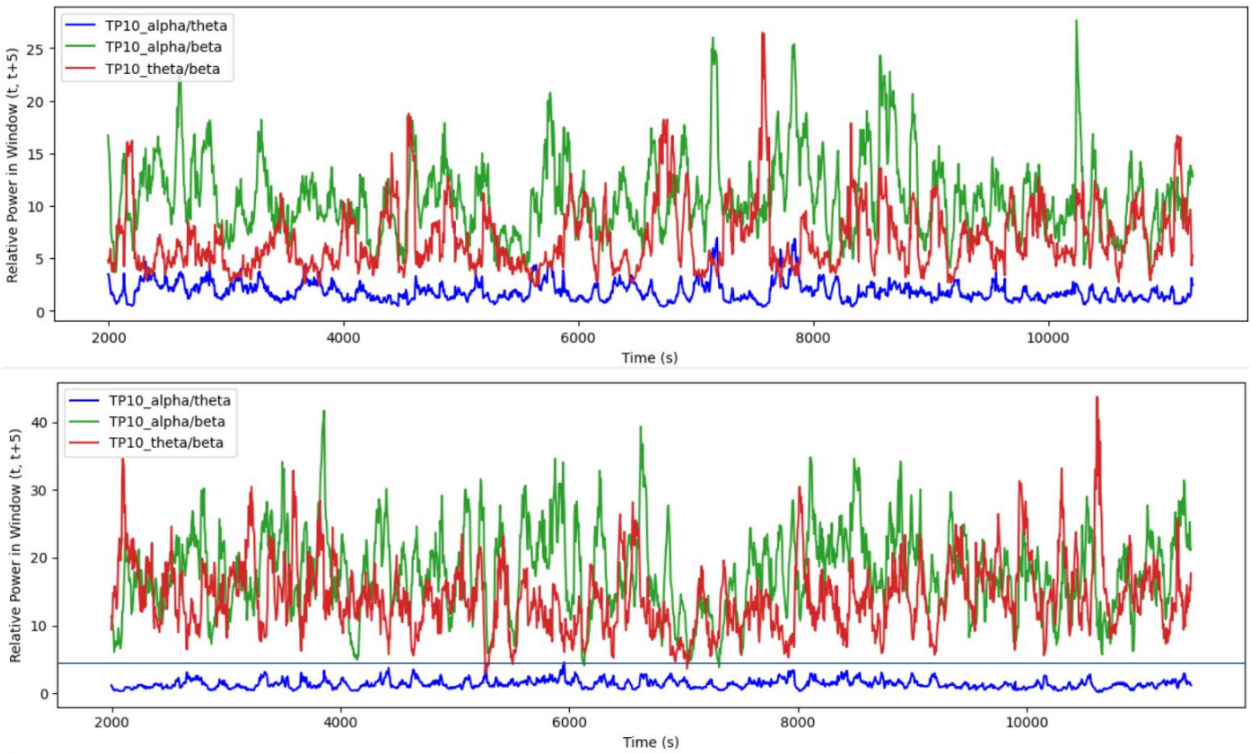


Figure 19: Awake (Upper) vs Drowsy (Lower) in Mode 3 of the Analysis (TP10 electrode)

In Figure 4, a test is shown with an awake sample on top and a drowsy sample below. Both were run through signal processing in mode 3, so the plot includes the alpha/theta, alpha/beta, and theta/beta power ratios. Also, both plots were created with a window of 1280 samples and an increment of 100 samples, and the original data came from the Muse (256 Hz). As expected, the alpha/beta and theta/beta ratios increased from the awake to drowsy state (note: the plot scaling is different). This is because the lower fatigue oriented frequencies, alpha and theta, increase in prominence and the focus-based higher frequency, beta, decreases. Also, the alpha/theta ratio decreased from awake to drowsy. This can be explained as the deep-fatigue oriented theta waves increasing more than the relaxation oriented alpha waves, as expected for a case of extreme exhaustion like in Figure 4.

As a final validation for the mathematical process in the recording analysis, the program was used to train the machine learning subsystem with both our own and online datasets, and the classification results from the trained model were promising, especially on the online dataset. As the live analysis implements the same calculation process, it is expected that the results will be similar after full integration next semester.

The live analysis functioned continuously for 40 minutes with the muse stream input during a test, and during the demo, it functioned perfectly during the entire demo time in each of the three output modes. None of these tests ended in failure, the program was just stopped. It was also validated to be resistant to stream interruption both before and during the demo. The program simply waits for the stream to resume and continues without crashing. In the most computationally extensive mode, mode 3, the computation time did not exceed 61 ms with 5 second intervals at 256 Hz (Muse) during a 20 minute test. Excluding 3 outliers, the max time was under 30 ms, so the program can handle much shorter intervals without synchronization problems.

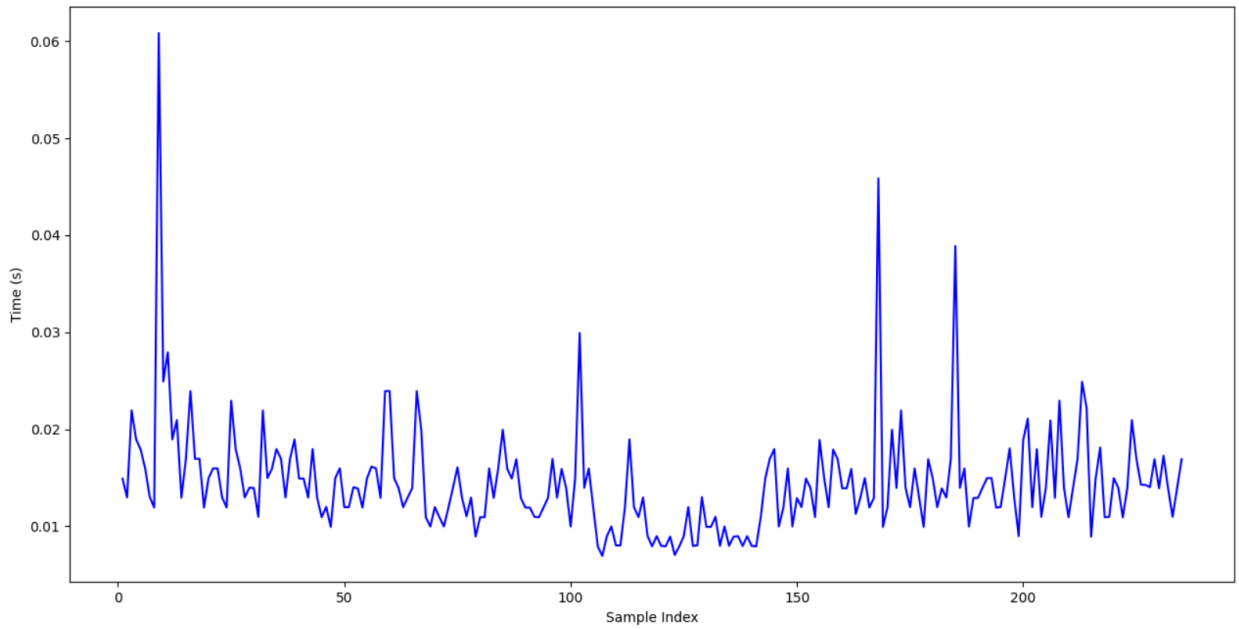


Figure 20: Computation Times on 5s Intervals at the Muse Frequency (20 minute test)

A testbench was created to run the live analysis on a data stream updated at 0.5 second intervals with a sampling rate of 1000 Hz, and it ran without issue. The code for the testbench is included. This spec was significantly faster than our validation plan spec of handling 10 second intervals in a livestream at 256 Hz without a loss of synchronization. The live analysis will have no problems with integration, as the EEG will not approach the speed of the testbench. Screenshots of live analysis running on both the Muse stream and the testbench are included below in figures 6 and 7, respectively. Both are of the most computationally difficult mode, and the final iteration, mode 3.

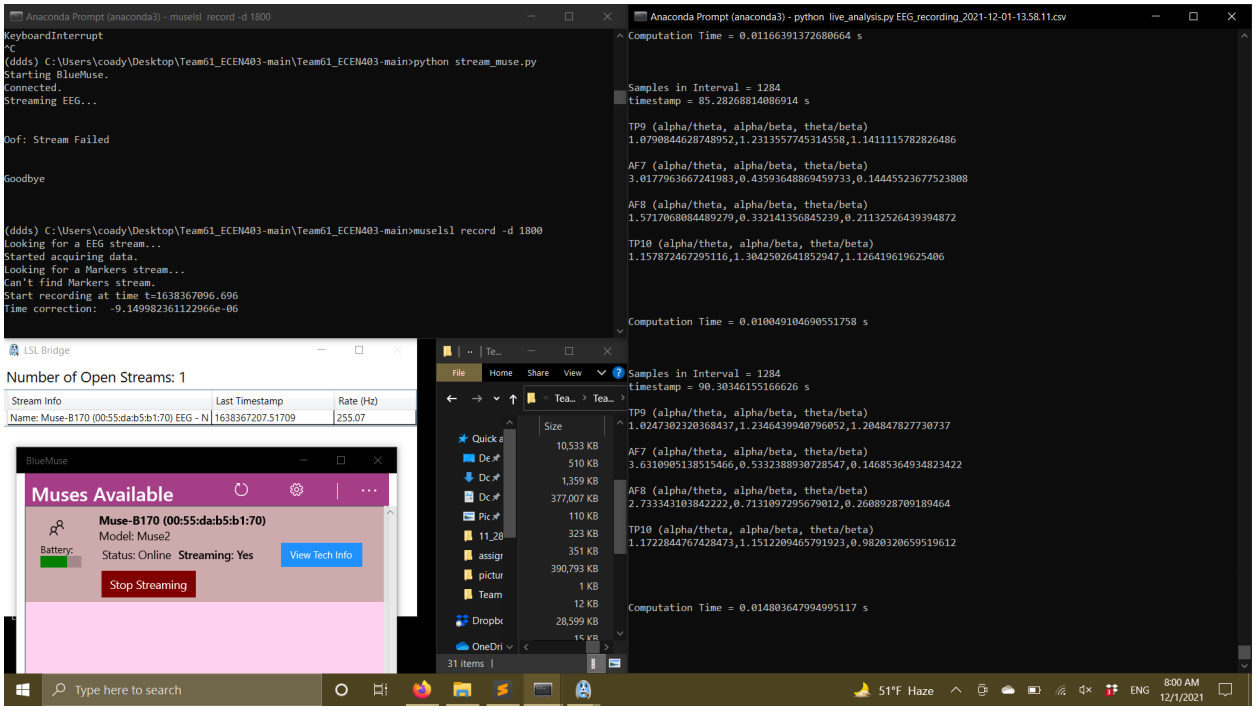


Figure 21: Live Analysis Running on the Muse Stream (5s Intervals, 256 Hz)

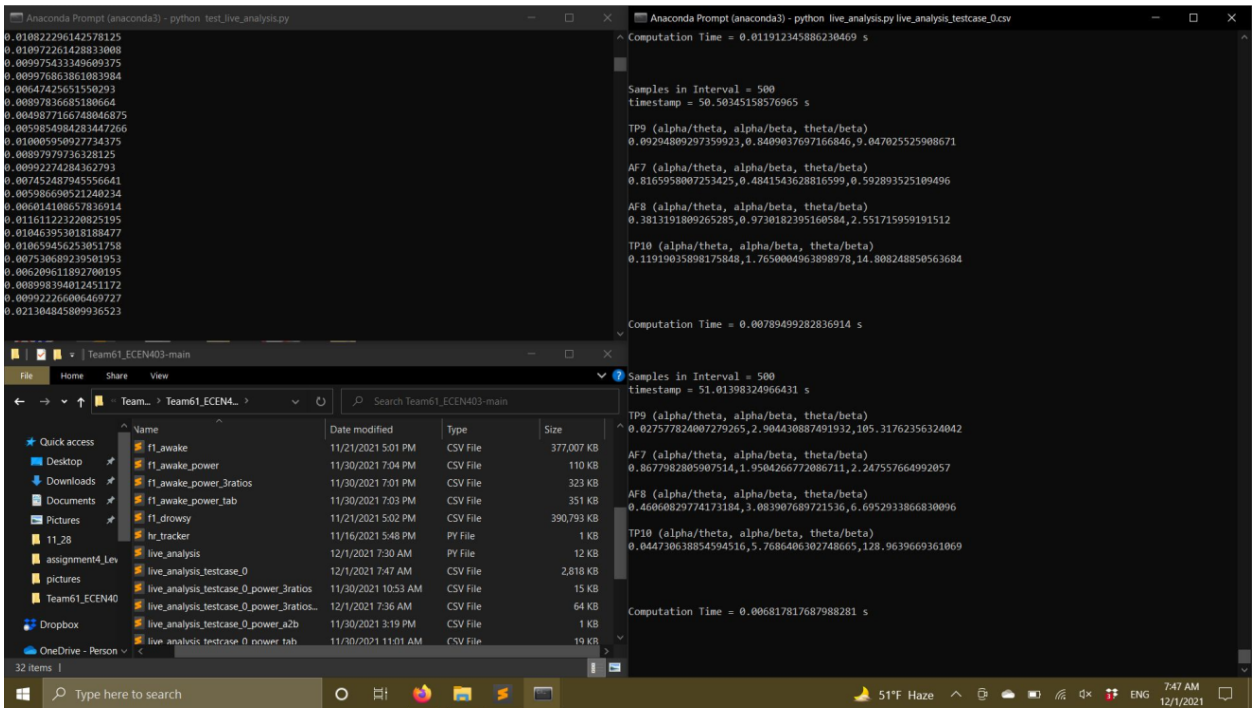


Figure 22: Live Analysis Running on the Testbench (0.5s Intervals, 1000 Hz)

4.4. Conclusion

The signal processing subsystem has exceeded the validation plan specifications on computation time. Additionally, the results of the machine learning model using this subsystem's outputs are very promising moving into 404. Partial integration has already been achieved, and moving forward the team will integrate live analysis with the machine learning model to perform real-time fatigue state classification with the Muse device, after that is completed, the team will move to replace the Muse with the EEG subsystem. In the time leading up to integration, the Raspberry Pi will be handled and programmed as part of this subsystem.