# Driver Drowsiness Detection System

Dakota Mouton, Ali Imran, Coady Lewis

## FINAL REPORT

30 April 2022

# FINAL REPORT
FOR
# Driver Drowsiness Detection System

TEAM 61

APPROVED BY:

_____
Project Leader Date

_____
Prof. Kalafatis Date

_____
T/A Date

# Table of Contents

# List of Figures

# Driver Drowsiness Detection System

Dakota Mouton, Ali Imran, Coady Lewis

# Concept of Operations

REVISION – II
30 April 2022

# 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 |
| II | 4/30/2022 | Ali Imran | | Hardware System (EEG and MCU) and Software System (ML and Signal Processor) more clearly differentiated |

# 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 two finished systems. First, we will create a trained model that will detect fatigue and output the user's current and recent states. Second, we will build a custom electroencephalogram (EEG). The model will accurately report the fatigue level of the user and plot their states in real-time. The software will also take logs of the user's output. This will provide trucking insurance companies necessary information to keep their drivers safe.

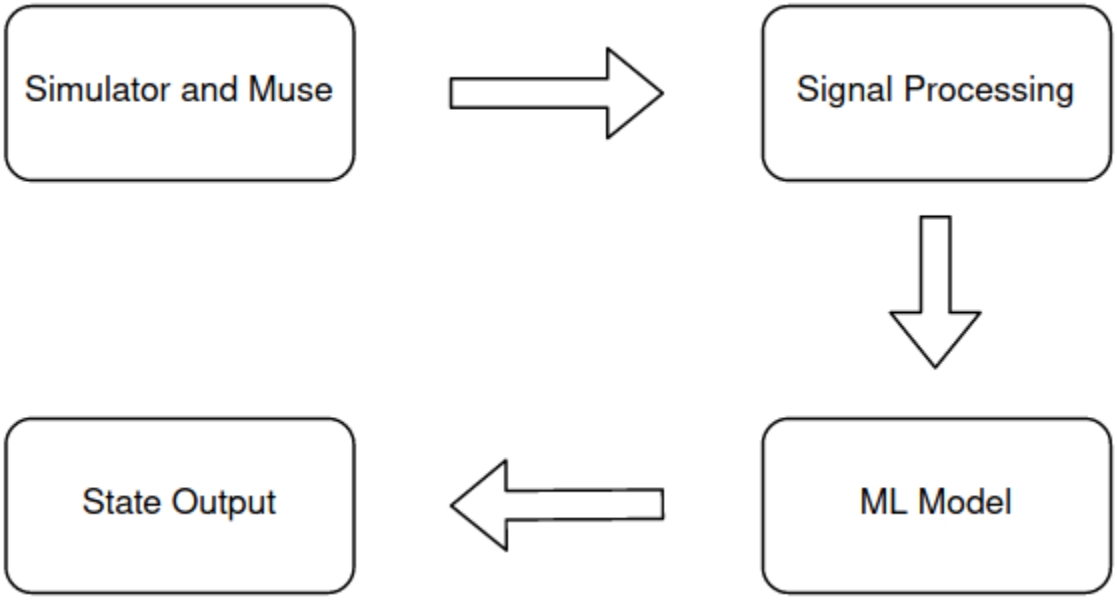



*Figure 1: Driver Drowsiness Detection Software Block Diagram*

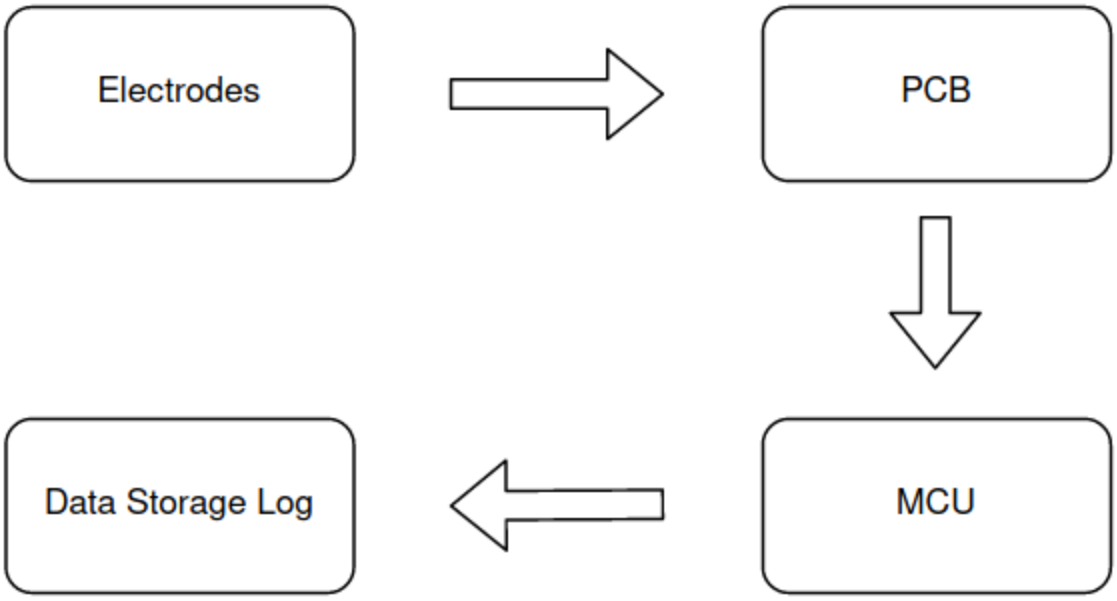



*Figure 2: Driver Drowsiness Detection Hardware Block Diagram*

# 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 giving the user a visual representation of their level of alertness. The DDDS will serve as a tool to inform drivers, 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 model will be used to monitor brain waves to detect an increase in fatigue. We will detect these brain waves using a Muse 2 EEG device. 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. The model will give a visual and terminal output, along with a datalog in csv format. The custom EEG will attempt to replicate the Muse output. The information captured by the EEG device will be fed through an analog circuit designed to cancel out excess noise. A microcontroller will then process this data so it can be recorded.

## *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 ultimately lead to a method for both trucking companies and insurance companies to measure fatigue for each individual driver. EEG signals will be collected from a band on the driver's head. 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 output accordingly. The model 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. The goal of the custom EEG is to attempt to produce a cheaper version of the Muse 2 commercial EEG.

## *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 the driver's 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 EEG will not function well in extreme heat.
- The model 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. 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.
- Live Signal Analysis: This system will perform feature extraction from the erratic raw EEG signals in order to train a more accurate model based on the frequency spectrum of the signals.

## *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 give the user live output of the specific level.

- **Awake (training label=0) -** When the user has what is considered normal brainwave activity being associated with full awareness and focus.
- **Near Awake (training label=0) -** When the user is not fully alert, but is still safe to drive. This is a transition state with weaker confidence in the model's output.
- **Near Drowsy (training label=1) -** When the user is very tired and entering a dangerous state. This is a transition state with weaker confidence in the model's output.
- **Drowsy (training label=1) -** When the device detects the level of brain activity that is below the drowsy threshold indicating a dangerous level of fatigue.

## *3.5. Users*

**Drivers -** The truck drivers can use the model 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 EEG can sit on their head 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 liability 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 documentation 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 an EEG band similar to the Muse that the model will be developed with. The EEG device will record the brain waves of the operator and send that information to the microcontroller. Given raw signals, the software will show an output describing the state of the driver. This output and the brain wave data could then be logged on a laptop which is then sent to the respective insurance companies. Once the driver is detected to be drowsy, the visual aid will show their state change in action

### *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 model to monitor their fatigue.

## 5. Analysis

### *5.1. Summary of Proposed Improvements*

- Since the DDDS is meant for final implementation in a portable EEG, 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 software 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 custom 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 – II
30 April 2022

# 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 |
| II | 4/30/2022 | Ali Imran | | Hardware System (EEG and MCU) and Software System (ML and Signal Processor) more clearly differentiated |

# 1. Introduction

## *1.1. Purpose and Scope*

The driver drowsiness detection system is made up of two systems. The Hardware System includes a custom 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. The Software System 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 COTS 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 |

### *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 | Rev. C | AD8221 Precision Instrumentation Amplifier Datasheet |
| 4 | LT 0314 REV.B | LTC2471/LTC2473 Datasheet |
| 5 | REC.1 | ADA4610 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 a Muse 2 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, simulator, a live signal analysis program 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. The EEG will send the raw brain activity to the microcontroller which will then send that information to a computer to record 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 a microcontroller to determine one’s current state without the need of a separate computer.

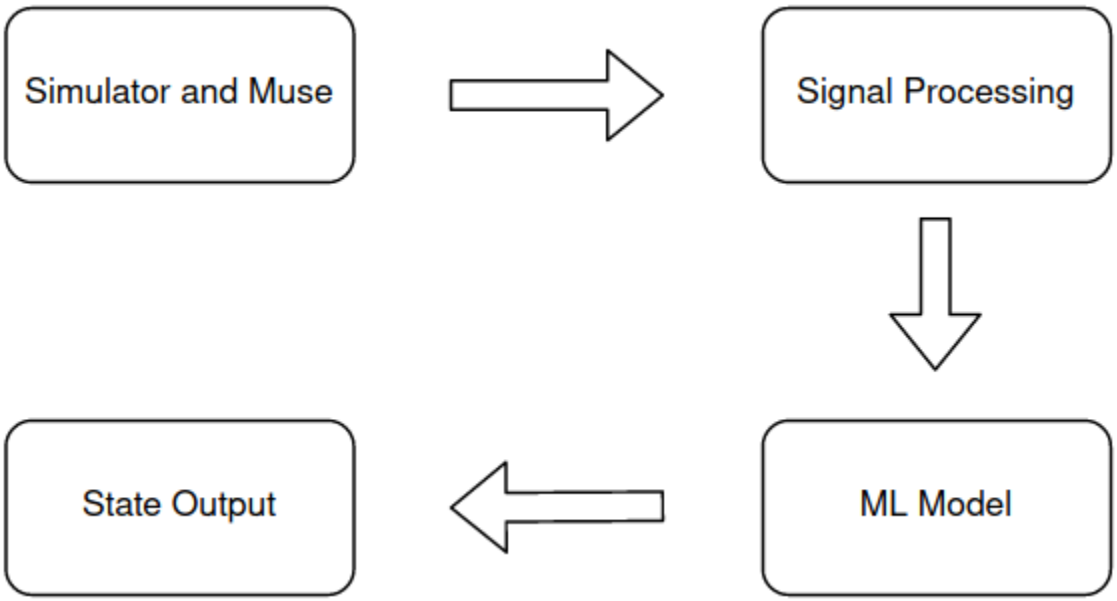



*Figure 3: Driver Drowsiness Detection Software Block Diagram*

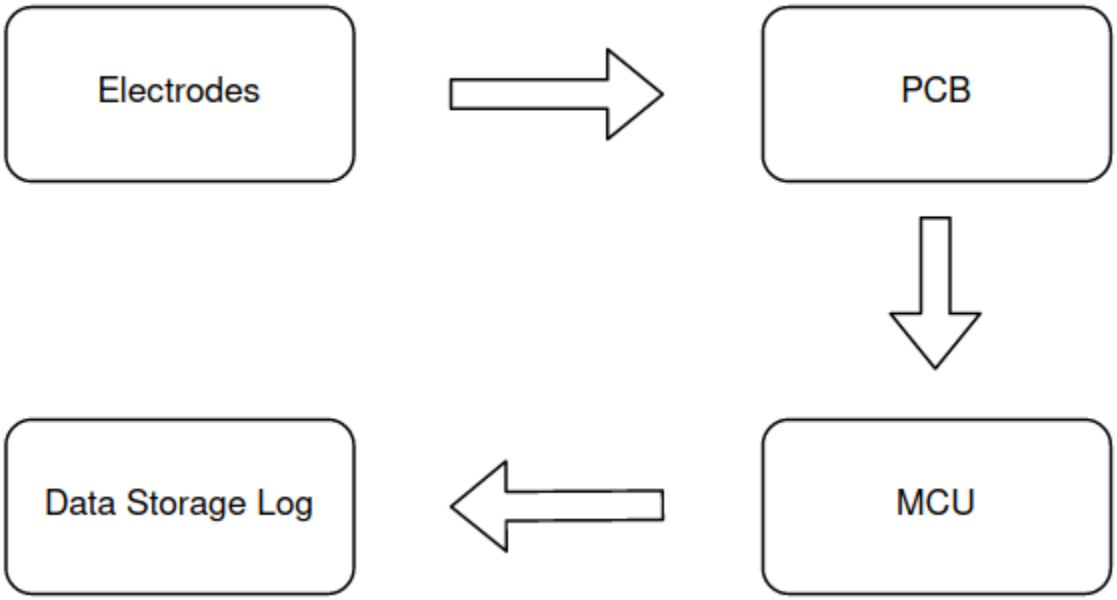



*Figure 4: Driver Drowsiness Detection Hardware Block Diagram*

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.

## *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 Arduino Uno R3 with external ADC will sample the EEG outputs at a maximum of 100 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 drowsiness in full awake and full drowsy states with at least 90% accuracy. It will also demonstrate the transition between the two states over time.

*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 custom EEG will last approximately 8 hours of continuous operation on four 1.5V batteries. The model with the muse will last approximately 4 hours of continuous operation.

*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 3.7.4. Python has many data science, signal processing, and machine learning tools that are free, easily accessible and syntactically simple to use.

#### 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.

#### 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. In sequential mode, the model will take the last 12 intervals as input and shift the sequence along by one interval every 5 seconds.

#### 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 millivolts for the brain activity, the Arduino Uno R3 will take those voltages through an ADC and send them to be recorded

##### 3.2.4.1.1 Power Consumption

a. The maximum peak power of the EEG 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 EEG System shall be 5V. The input voltage level for the MCU shall be 5V.

*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 software shall include a graphical output 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 software shall include a terminal output for error detection.

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

#### 3.2.4.3. Wiring

The EEG shall follow the guidelines outlined in IEC 60601-1:2012.

*Rationale: Conform to medical device standard.*

### 3.2.5. Environmental Requirements

The EEG 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 EEG 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 EEG 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

The EEG 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 EEG 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 software will alert the driver when the model is no longer working. This will be resolved either by charging the Muse or laptop battery, resetting the stream, or contacting us for troubleshooting options.

# 4. Support Requirements

The customer just needs to be able to change out the batteries when the EEG device is dead, as well as charge the Muse if using it for training.

# 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 – II
30 April 2022

# 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; |
| II | 4/30/2022 | Ali Imran | | … |

# 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.

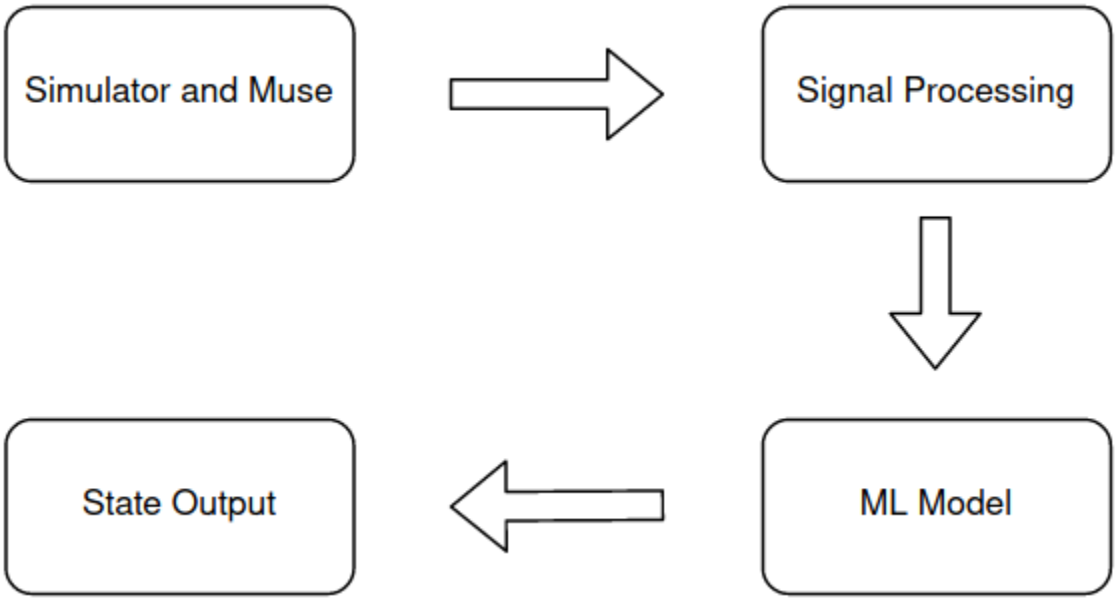



*Figure 5: Driver Drowsiness Detection Software Block Diagram*

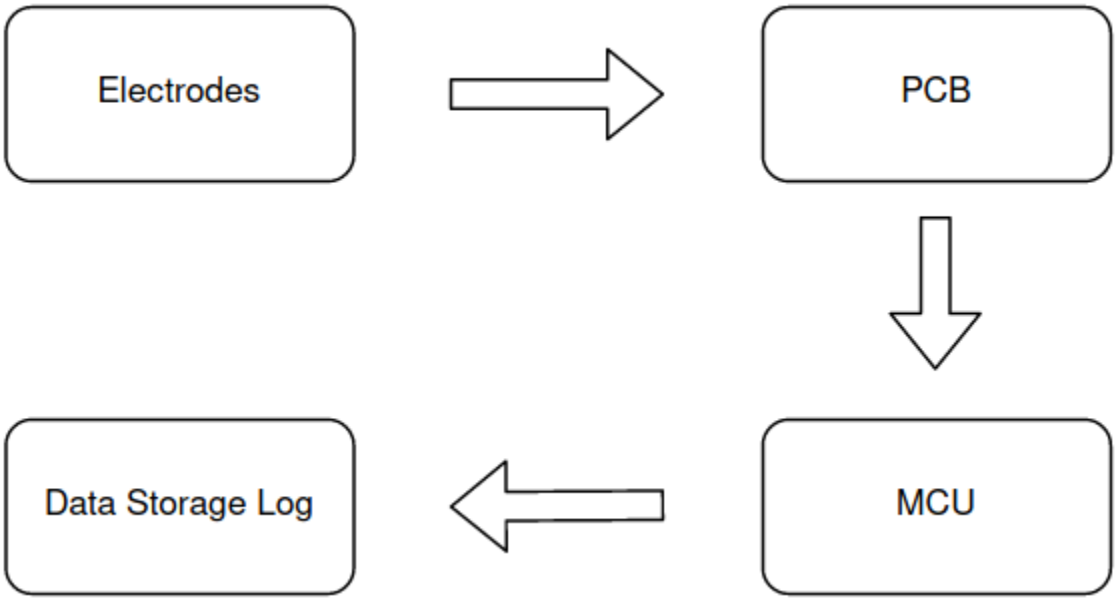



*Figure 6: Driver Drowsiness Detection Hardware Block Diagram*

# 2. References and Definitions

## *2.1. References*

- IEC 6061 Technical standard for medical electrical equipment
- IEEE Recommended Practice for Neurofeedback Systems

## *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

25 grams

## *3.2. Dimensions*

3.2.1 EEG

15cm X 4cm

3.2.2 MCU

53.4 mm X 68.6 mm

## *3.3. Mounting Locations*

3.3.1 EEG

The EEG will have wearable electrodes to maintain contact with the user’s 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, total of 6V, 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 USB via computer at 5V.

## *5.2. Voltage and Current Values*

| Component | Voltage | Current |
|---|---|---|
| EEG | 5V | 500 mA |
| MCU | 5V | 200 mA |

## *5.3. MCU*

We will be using the Arduino Uno R3 for this project. This device was chosen based on its large library selection and wide range of projects to refer to when writing the code for its operation. It is also affordable and easy to use compared to a custom MCU solution.

## *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 16 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 I2C.

# 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, Kernel SVM, Recurrent Neural Network, and a Recurrent Neural Network with Convolutional layers.

Driver Drowsiness Detection System Schedule (403):

| 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 (403):

| Task | Specification | Result | Owner |
|---|---|---|---|
| ML Algorithm drowsiness detection | >90% success rate | 85% w/ collected data<br>97% w/ online data | Ali, Coady |
| Performance of data collection and processing every interval:<br>● EEG data collection<br>● Transfer and signal processing<br>● Fatigue state output (ML model) | < 30 seconds<br>< 10 seconds<br>< 10 seconds<br>< 10 seconds | Integration Needed<br><br><61 ms<br>(0.005 - 0.05 s) | All<br>Dakota<br>Coady<br>Ali |
| EEG Filters below 8 Hz | f>=8 Hz | Pass, See screenshots | Dakota |
| EEG Filters Above 30 Hz | f<=30 Hz | Pass, See screenshots | Dakota |
| Final voltage readings have amplified from millivolts to volts | 0.148<V<0.81172<br>But V<1.25<br>Input 20 Hz, 160mV | Vrms = 0.809 V | Dakota |
| System voltage input | 6 V | Regulated to +-5V | All |

# Driver Drowsiness Detection System

Dakota Mouton, Ali Imran, Coady Lewis

# Subsystem Report

REVISION – I
30 April 2022

# 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 |
| I | 4/30/2022 | Entire Team | | Subsystem Changes for 404 |

# 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.

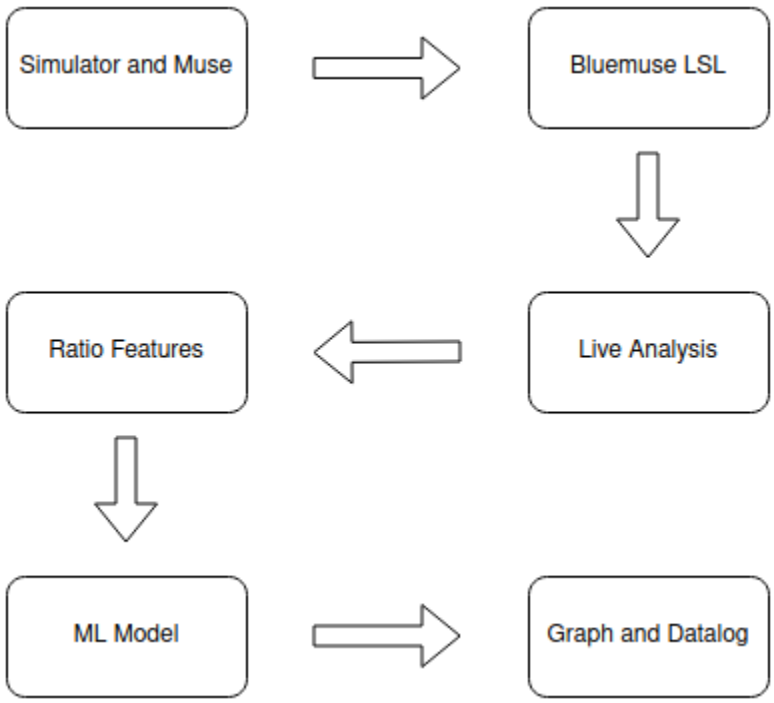



*Figure 7: Driver Drowsiness Detection Software Block Diagram*

# 2. EEG Device Subsystem

*Changes will be marked with an asterisk at the start of the change.

## 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, and the circuit should filter out everything else. 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

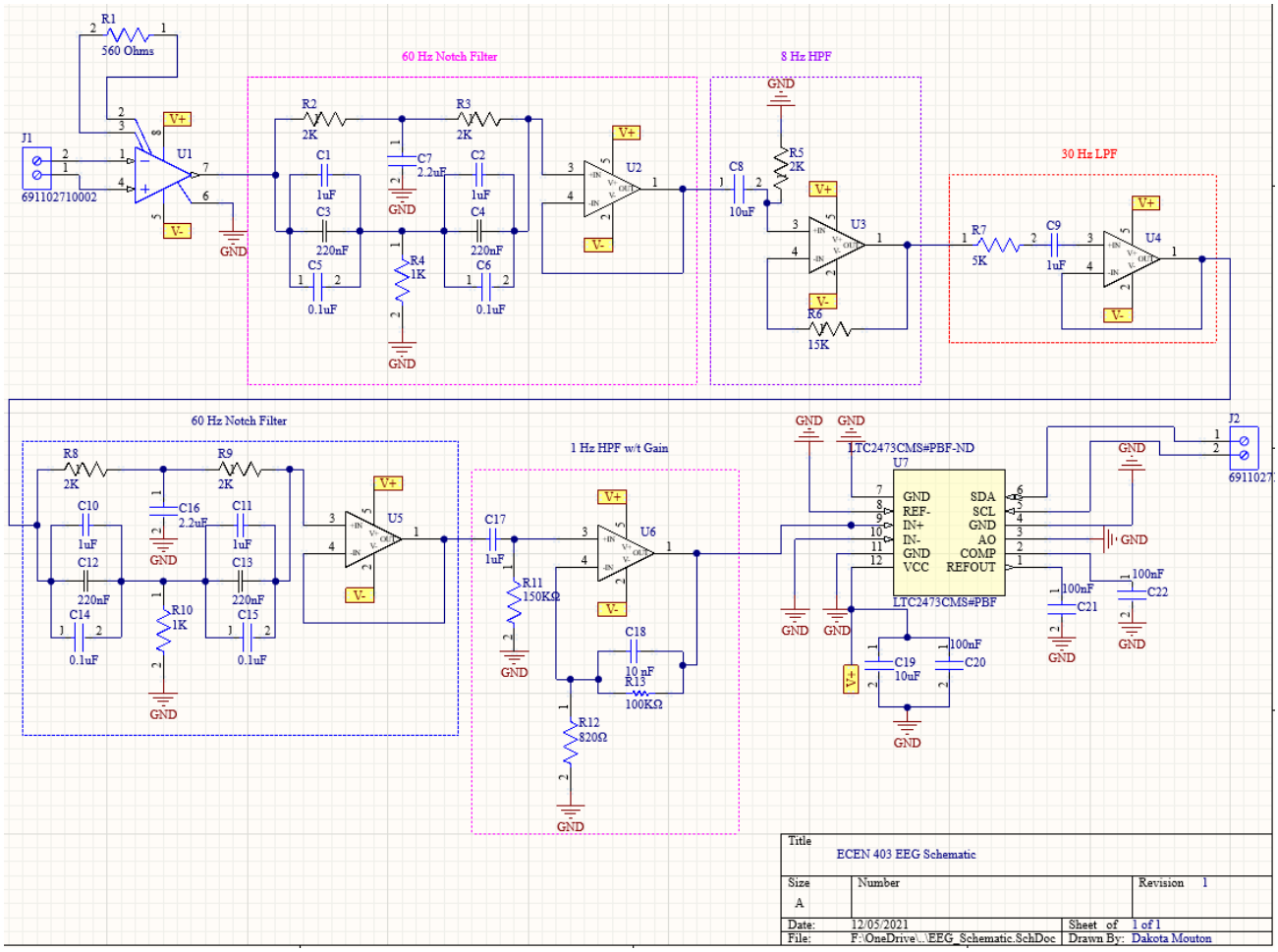



**Figure 8: EEG Circuit V1**

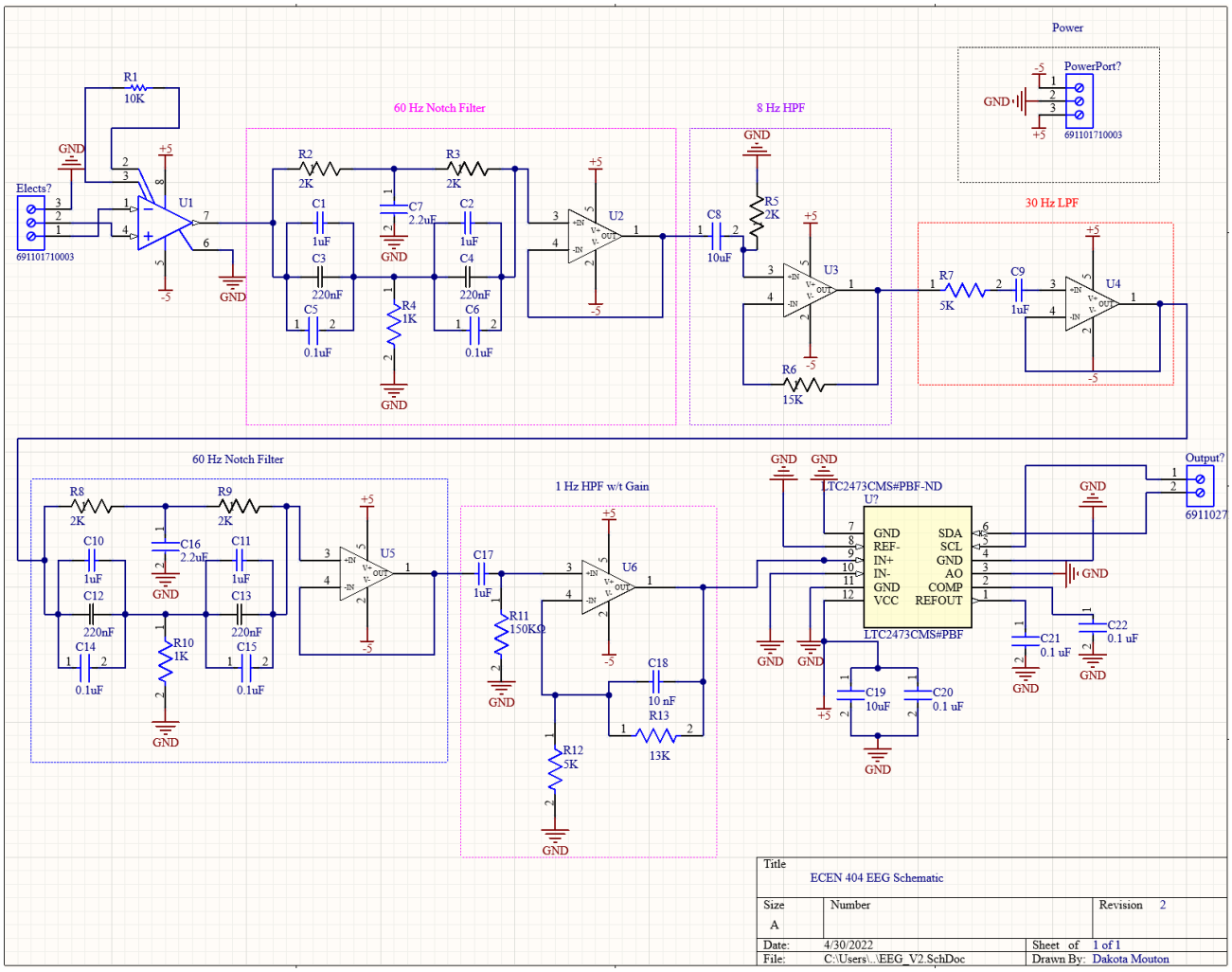



**Figure 9: Finalized EEG Circuit V2**

*The EEG circuit shown above had some slight changes made to it at the end of the project timeline. The original design in Figure 2 has a 560Ω resistor for R1 to have a gain of 89.2 V/V to amplify microvolt input signals into the milivolt range for further filtering and processing, however we found that our input electrodes would pick up lots of interference, especially in the EDC lab, all the way up to a volt but more consistently anywhere between 30 mV to 160 mV, which means our gain was way too high and was actually clipping the input signal and resulting in a very high voltage of around 4.3V on the output of just U1 and saw almost 0V on the rest of the op amp filters. To resolve this problem we swapped R1 with a 10kΩ resistor which brought the gain down to 5.94 V/V and with our simulated input of 160 mV to around 672 mV on the output of the instrumentation amplifier. Further results will be shown in the validation section. Two more resistors, R12 and R13 were swapped out to 5kΩ and 13kΩ resistors, respectively, because the 1 Hz HPF was also designed to amplify the output one more time before being sent into the ADC, however the initial design still had to high of a gain which resulted in an output over the reference 1.25 V the ADC needed for conversion. By swapping out these resistors we were able to get the output below this threshold. We also had to make a small hack/fix on capacitor C9 as it was improperly hooked up directly between resistor R7 and the input of U3 which resulted in no filtering, so

we connected one end of the capacitor to ground and the other end, via a physical wire, in between R7 and U3 which then brought back the 30 Hz low pass filtering.

### 2.2.2 The Simulation Results

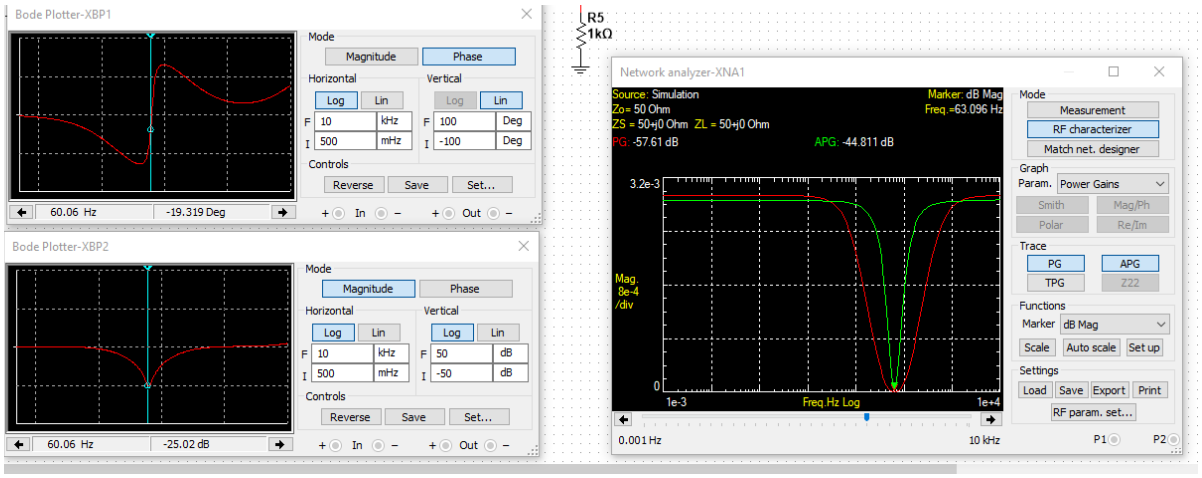



**Figure 10: 60 Hz Notch Filter Bode Plots**

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.

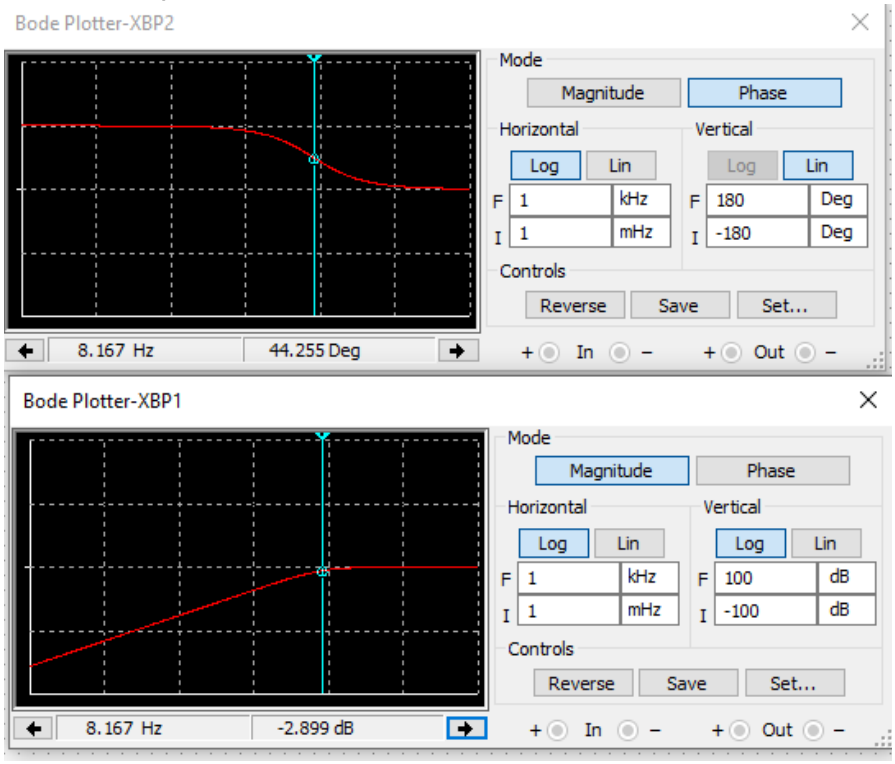



**Figure 11: 8 Hz High Pass Filter Bode Plots**

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.

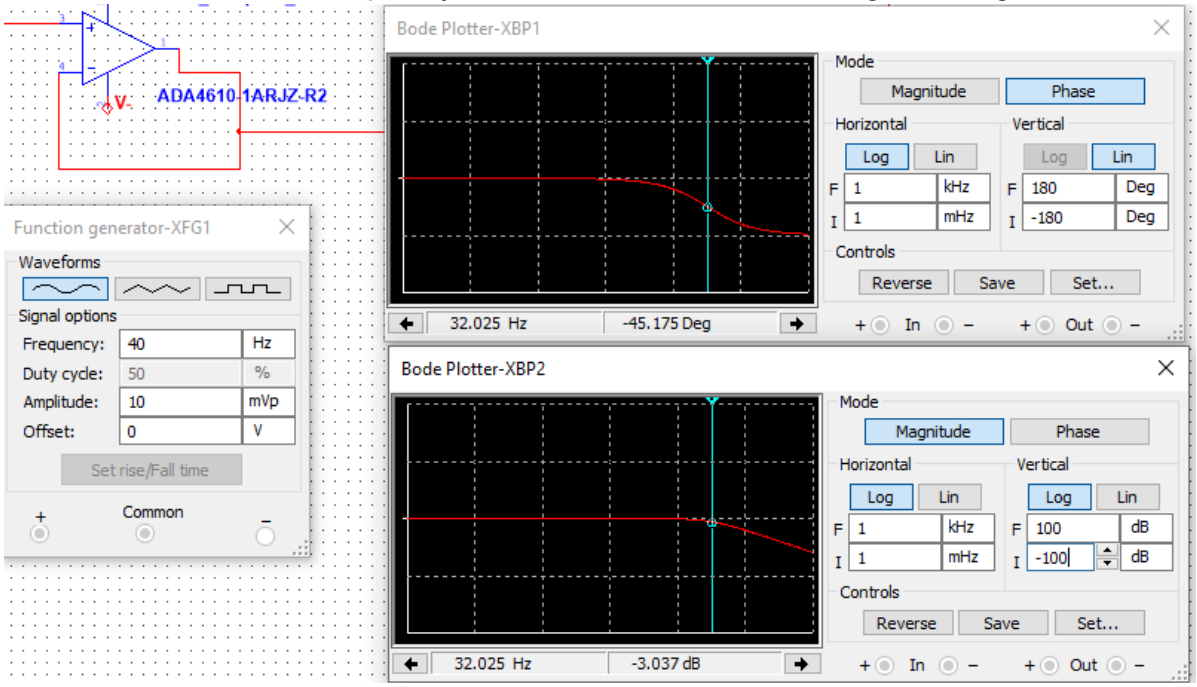



**Figure 12: 30 Hz Low Pass Filter Bode Plots**

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.

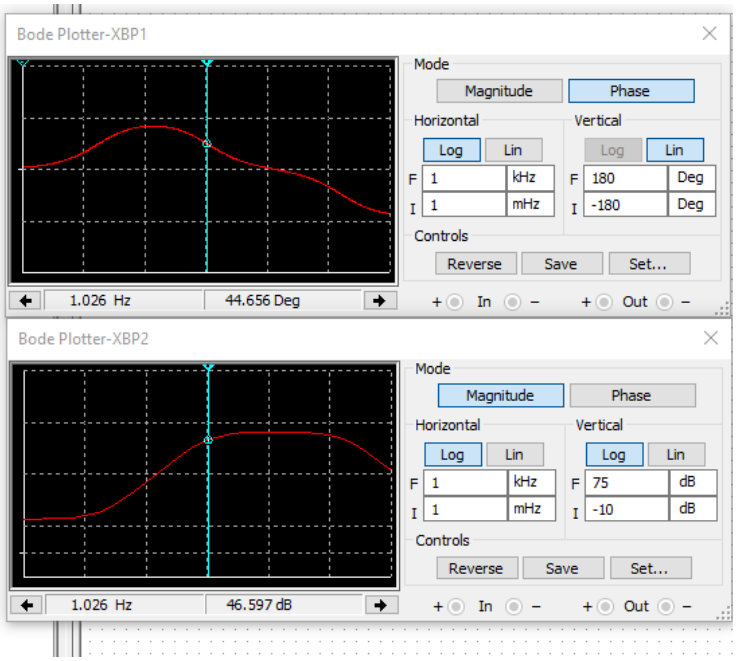



**Figure 13: 1 Hz High Pass Filter Bode Plots**

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.

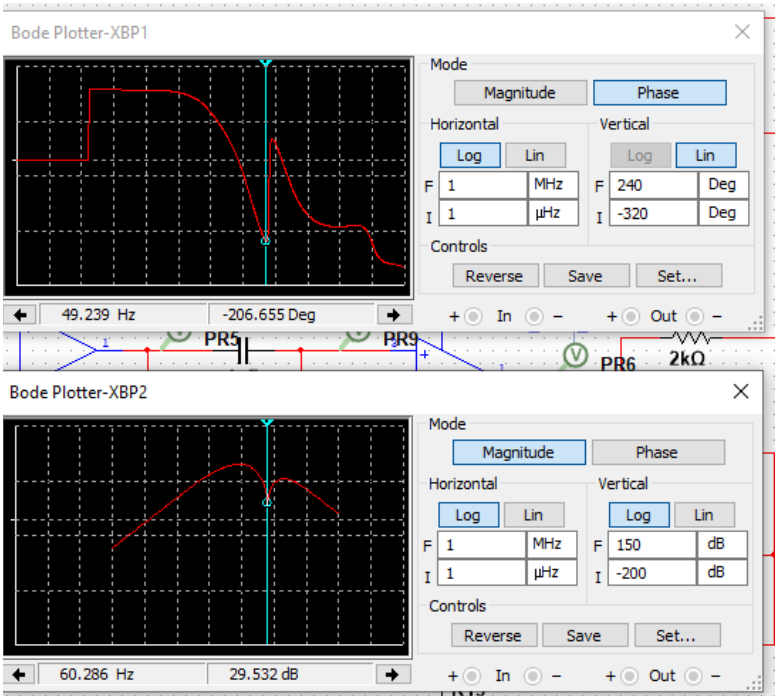



**Figure 14: Full EEG Output**

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.

### 2.2.3 The PCB Design

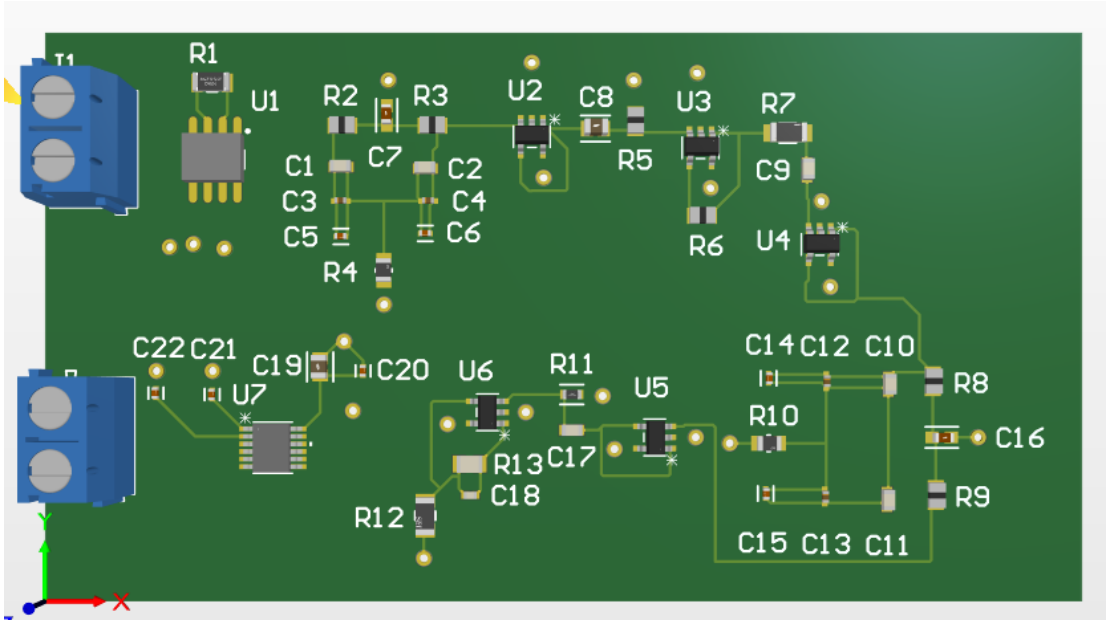



**Figure 15: PCB Design V1**

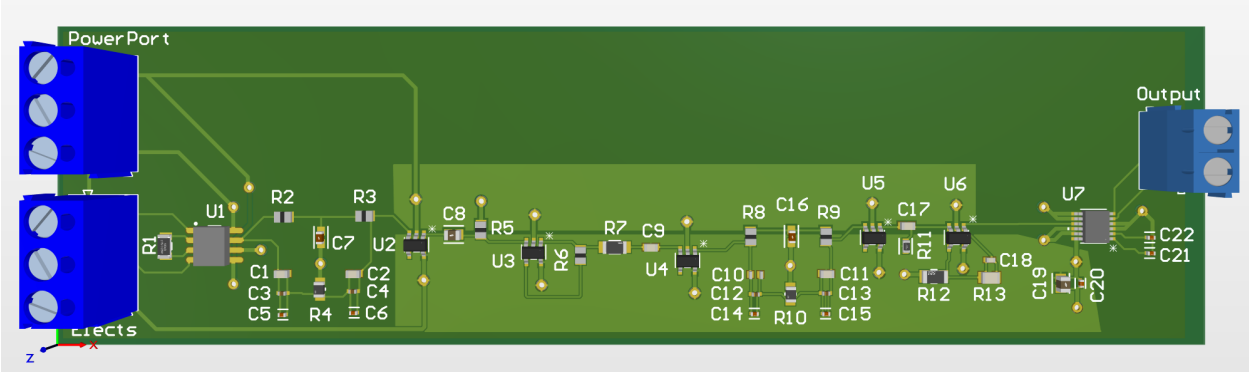



**Figure 16: PCB Design V2**

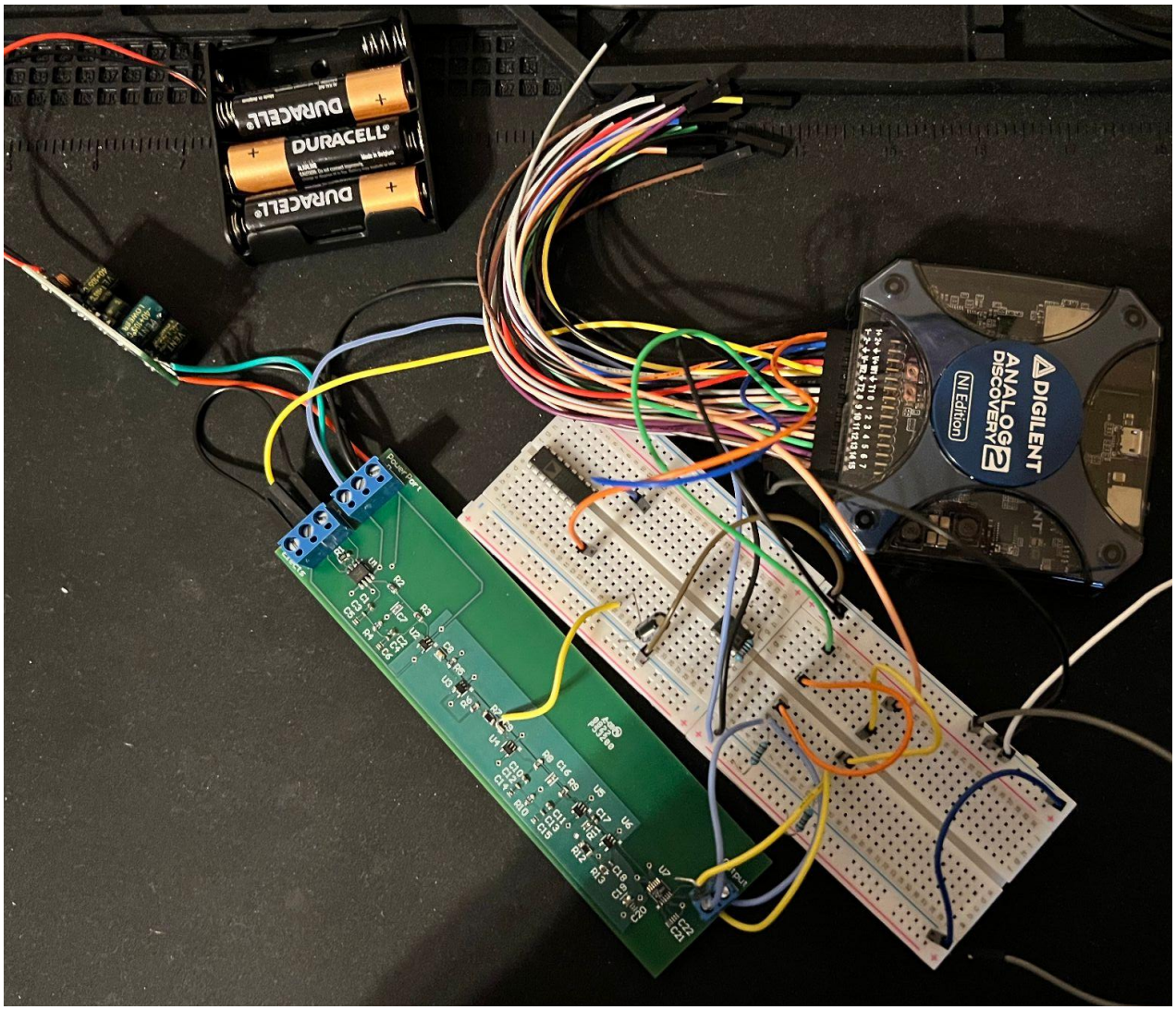



**Figure 17: Full PCB System**

*The initial PCB design found in Figure 8 was an early mock up of the positioning and orientation of the IC's in order to reduce the board's physical dimensions. However, for practical purposes and my personal novice soldering skills I ended up rearranging the components to be more elongated for better access to the soldering pads and also making sure components were not too close for potential signal interference. Although this redesign elongated the board, the purpose of this board was to test the concept of building an affordable, functioning EEG device. The results show great functionality of the board. In future iterations and testing of this design the physical dimensions could be improved and optimized to be as small as possible without causing too many drawbacks to the performance of the device.

*In Figure 10 we can also see the yellow wire towards the middle of the PCB which is the hack/modification I made to fix the 30 Hz low pass filter to actually filter the signal, otherwise it had no function other than passing the signal along. Also shown is the Analog Discovery 2 device which was used to input our simulated signal ranging from 20 mV to 160 mV and changing the test frequency to show the op amp filterings. We used this because the electrodes we initially wanted to use were picking up too much interference and causing very inconsistent input results which made validating the system almost impossible. Therefore by using the controlled simulated input from this device we were able to test a range of input voltages and frequencies to verify functionality.

## 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 | -19.319 | -25.02 | 1.57 |
| 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**

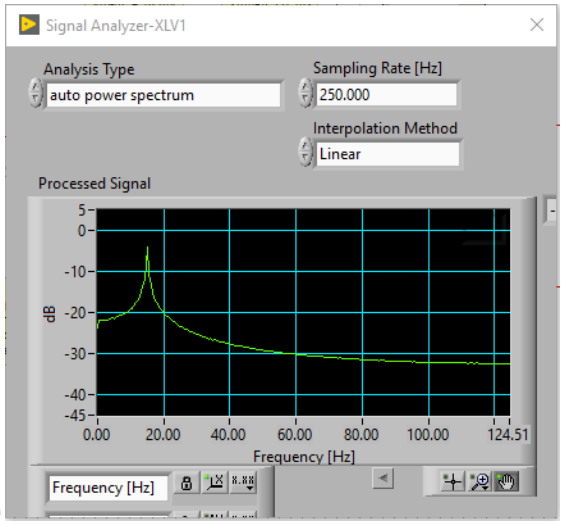
| Input Frequency: (Hz) | Input Voltage: (mV) | RMS Voltage: (mV) | Cut-off Frequency: (Hz) | Phase shift: (Deg) | Magnitude: (dB) | Output Voltage: (mV, RMS) |
|---|---|---|---|---|---|---|
| 10 | 0.02 | 0.0141 | 54.483 | -203.193 | 29.532 | 396 |
| 30 | 0.02 | 0.0141 | 54.483 | -203.193 | 29.532 | 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)**

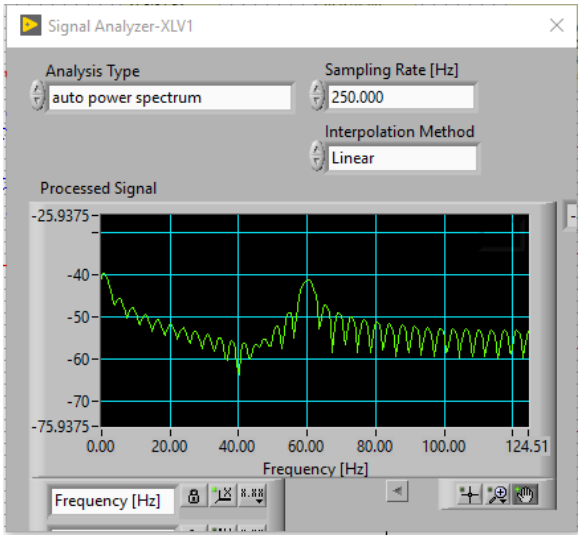
| Input Frequency: (Hz) | Input Voltage: (mV) | RMS Voltage: (mV) | Cut-off Frequency: (Hz) | Phase shift: (Deg) | Magnitude: (dB) | Output Voltage: (mV, RMS) |
|---|---|---|---|---|---|---|
| 15 | 0.03 | 0.0212 | 60.286 | -155.042 | 19.682 | 148 |

### 2.3.2 Simulation Graphs

a.)




b.)




c.)

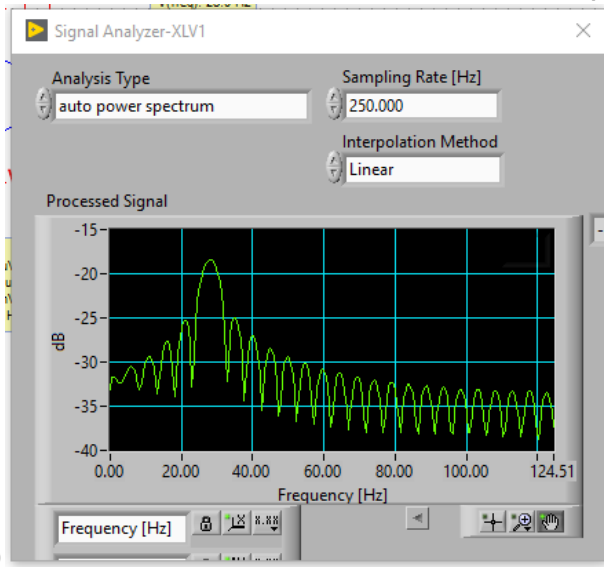



**Figure 18: a.) 15 Hz test input, b.) 60 Hz test input, c,) 30 Hz test input**

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 and Results

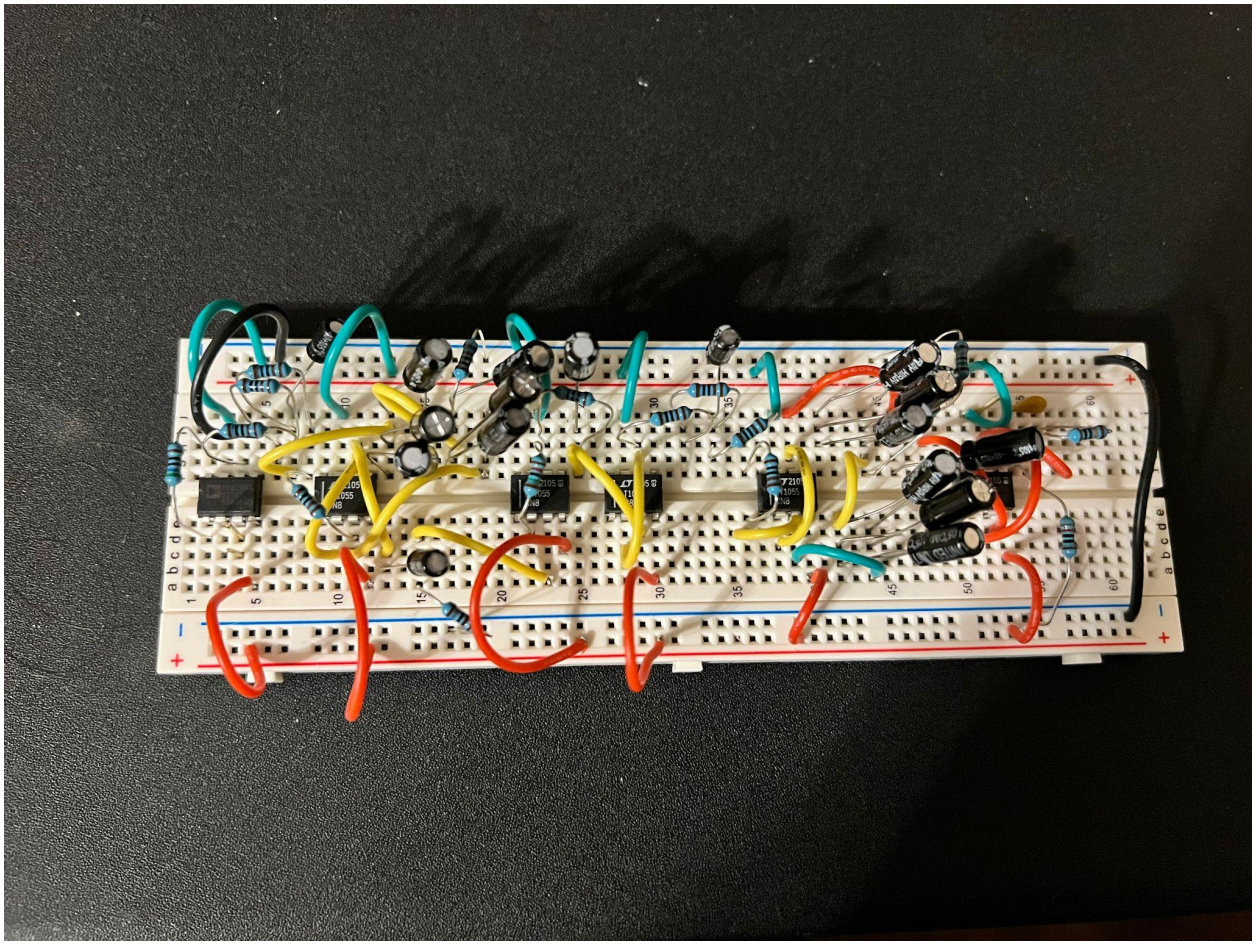


**Figure 19: Breadboard of EEG Circuit**

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.

*The breadboard proved to be a very unreliable form of testing the functionality of my EEG design as there was always inconsistencies or issues when trying to test, therefore results collected from this circuitry are basically meaningless and do not hold any real value

in the analysis of this circuit as my results were never consistent enough to have a clear idea as to what the circuit was doing.

### 2.3.4 PCB Tests and Results

| | | | |
|---|---|---|---|
| 1 | Positive Electrode Voltage: | 0.736 V | |
| 2 | Negative Electrode Voltage: | 0.756 V | |
| 3 | U1 Postivie Input: | 0.8 V | |
| 4 | U1 Negative Input: | 0.758 V | |
| 5 | U1 Output: | 0.590 V | |
| 6 | U2 Input: | 0.580 V | |
| 7 | U2 Output: | 0.576 V | |
| 8 | U3 Input: | 0.009 V | neg |
| 9 | U3 Output: | 0.008 V | neg |
| 10 | U4 Input: | 0.005 V | neg |
| 11 | U4 Output: | 0.065 V | |
| 12 | U5 Input: | 0.069 V | |
| 13 | U5 Output: | 0.171 V | |
| 14 | U6 Input: | 0.005 V | neg |
| 15 | U6 Output: | 4.3 V | |
| 16 | ADC Input: | 4.41 V | |
| 17 | SDA Line: | 3.75 V | |
| 18 | SCL Line: | 3.75 V | |

**Figure 20: PCB Voltages From Figure 2 Design**

| | |
|---|---|
| Test 2 (4-21-22) | |
| *input is 160 mV AC, 20 Hz from AD2 | AC RMS |
| U1 Postivie Input: | 113 mV |
| U1 Output: | 672 mV |
| U2 Input: | 414 mV |
| U2 Output: | 415 mV |
| U3 Input: | 362 mV |
| U3 Output: | 363 mV |
| U4 Input: | 360 mV |
| U4 Output: | 362 mV |
| U5 Input: | 231 mV |
| U5 Output: | 232 mV |
| U6 Input: | 232 mV |
| U6 Output: | 2.63 V |
| ADC Input: | 2.63 V |
| SDA Line: | 3.76 V |
| SCL Line: | 3.76 V |
| Refout | 1.25 V |
| Comp | 1.87 V |
| *Replaced R1 with 10K | |
| Gain of 5.94 V/V on U1 | |

**Figure 21: PCB Voltages From Figure 2 Design w/t R1=10kΩ**

| | | |
|---|---|---|
| *input is 160 mV AC, 20 Hz from AD2 | AC RMS | |
| U1 Postivie Input: | 113 mV | |
| U1 Output: | 672 mV | |
| U2 Input: | 412 mV | |
| U2 Output: | 413 mV | |
| U3 Input: | 361 mV | |
| U3 Output: | 362 mV | |
| U4 Input: | 360 mV | |
| U4 Output: | 362 mV | |
| U5 Input: | 230 mV | |
| U5 Output: | 231 mV | |
| U6 Input: | 232 mV | |
| U6 Output: | 809 mV | |
| ADC Input: | 809 mV | |
| SDA Line: | 3.76 V | |
| SCL Line: | 3.76 V | |
| Refout | 1.25 V | |
| Comp | 1.87 V | |
| *Replaced R12 with 5K and R13 with 13K and has R1 at 10K | | |

**Figure 22: PCB Voltages From Figure 3 Design**

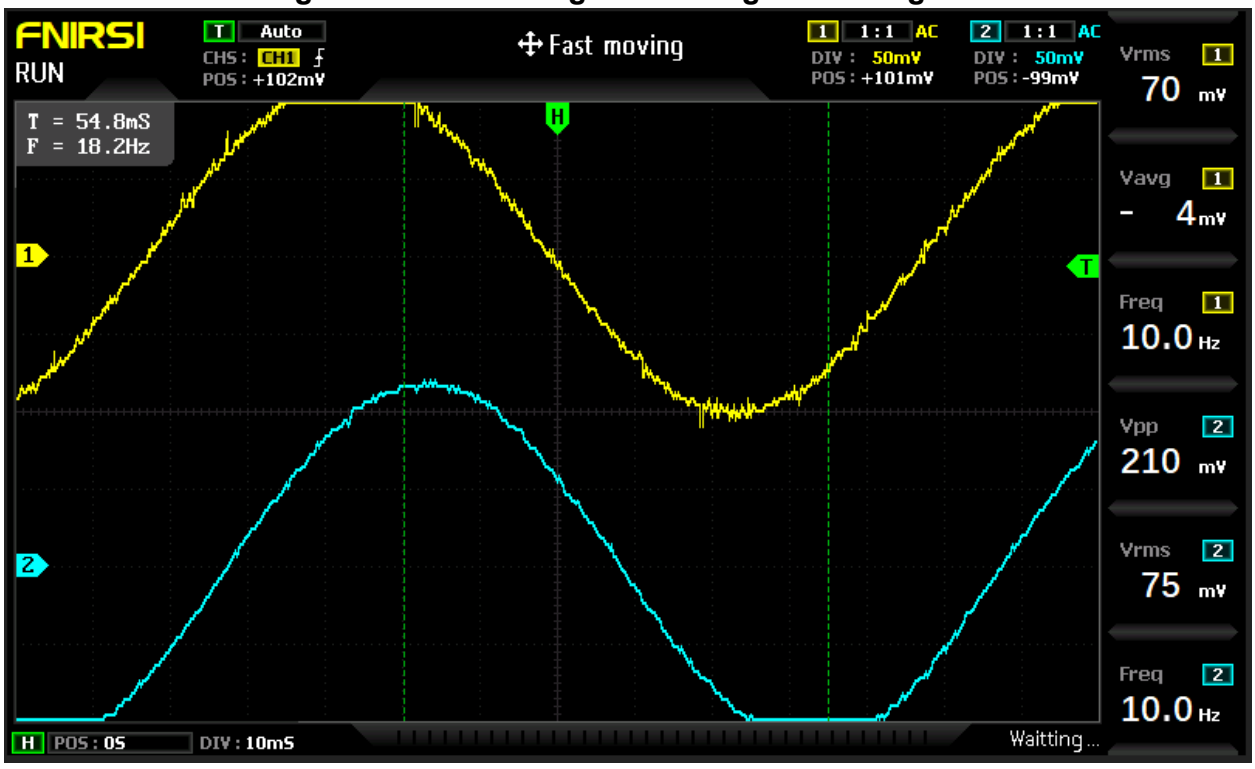



**Figure 23: Input vs Output of PCB (100mV, 10Hz input)**

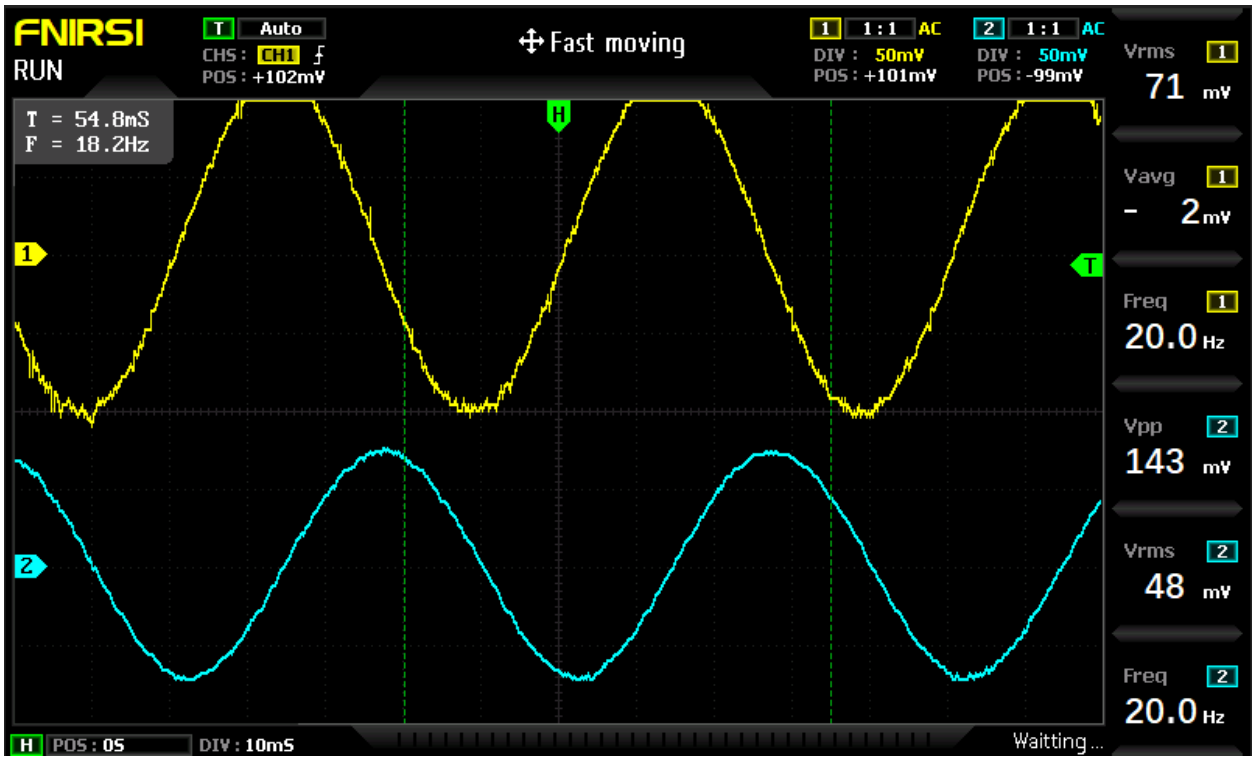



**Figure 24: Input vs Output of PCB (100mV, 20Hz input)**

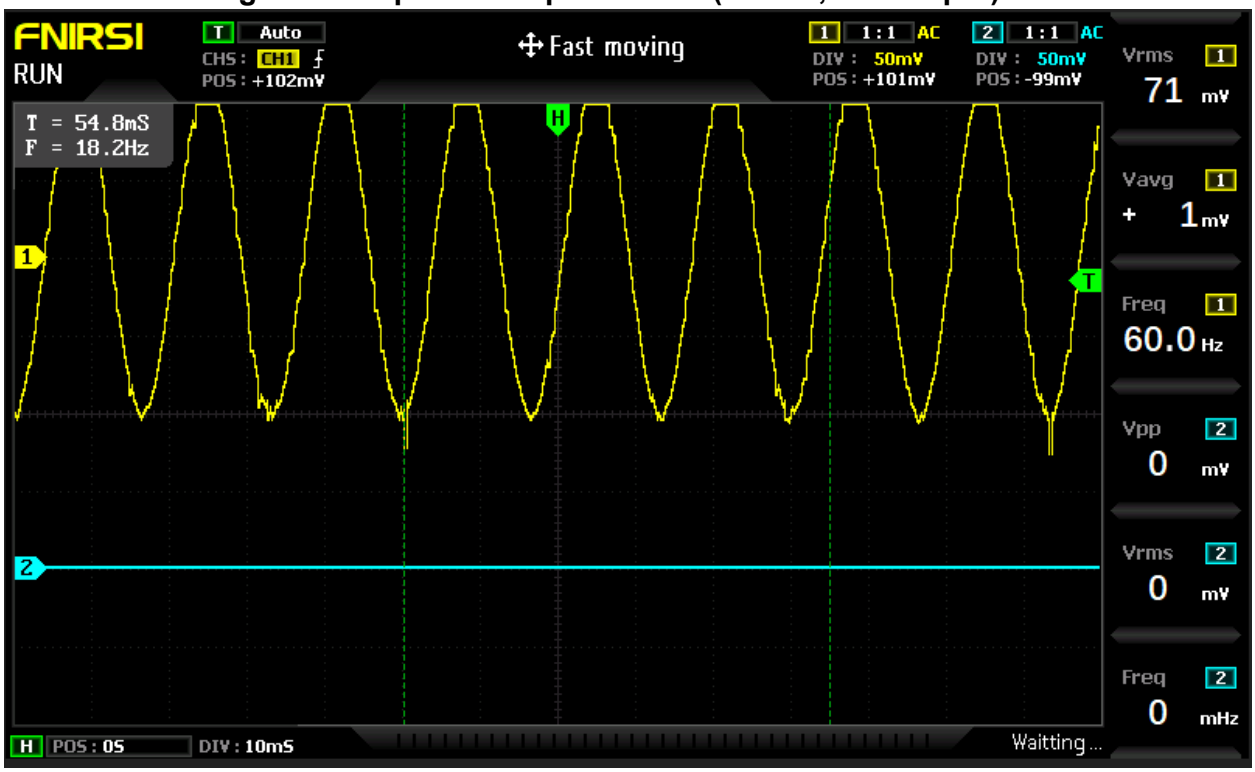



**Figure 25: Input vs Output of PCB (100mV, 60Hz input)**

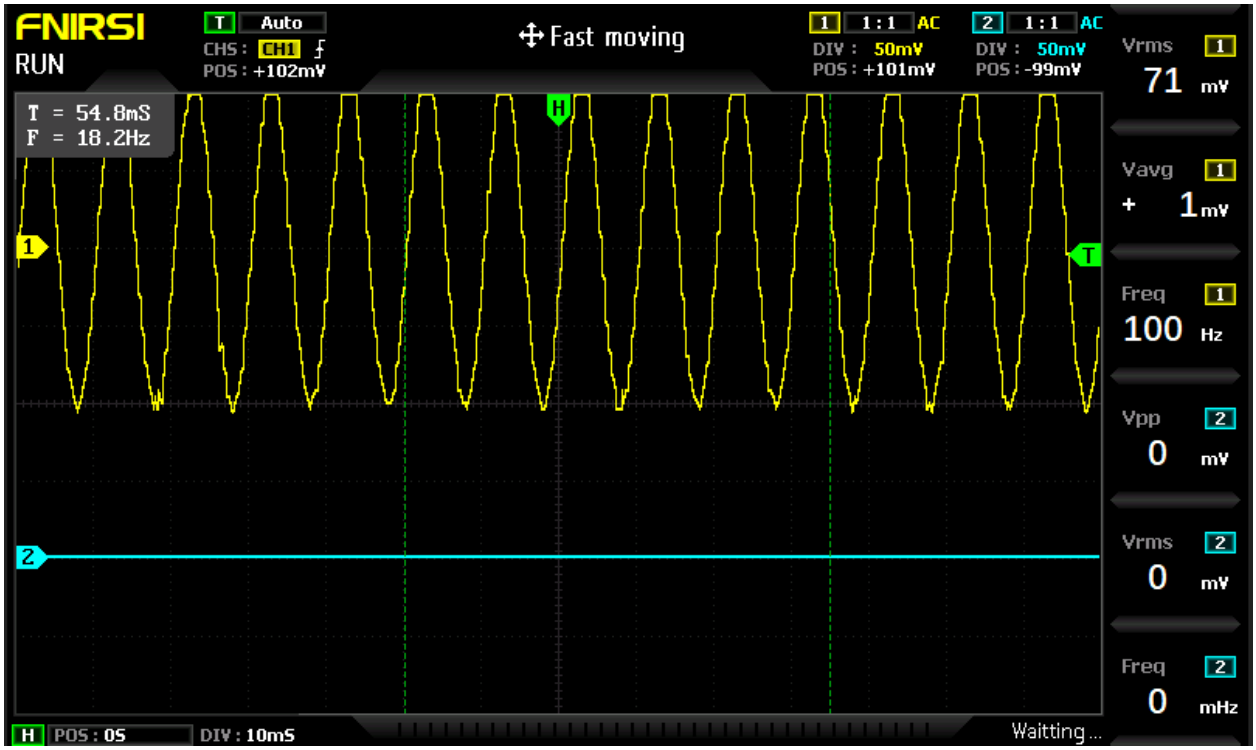



**Figure 26: Input vs Output of PCB (100mV, 100Hz input)**

Figures 11 through 13 show the measured voltages throughout the circuitry to show the effects my design had before and after changing out the resistors mentioned before, R1, R12 and R13. Figure 11 shows my initial measurements attempting to use the electrodes for measurements and validation. As you can see the input into each of the two electrodes was around 700 mV and the outputs very quickly diminished by the time it got to U3, the 30 Hz LPF. These measurements were all over the place and varied greatly depending on the location of my testing so these results were very inconclusive. However, by switching to the AD2 for simulated input at 160 mV, 20 Hz, we are able to see what the circuit is actually doing. Figure 12 shows the measured voltages on the circuit after adding in the 10kΩ resistor for R1. We can see from this figure that the input voltage ends up around 113 mV and the final output from U6, before going into the ADC, is 2.63 V, which is way too high for the ADC to do its conversions, so we needed to tone down the gain on U6. By swapping in the new R12 and R13 resistor values, we can see from figure 13 that the input was measured around 113 mV again for the same generated input from the AD2 but this time the output from U6 is 809 mV which is within our bounds and allows us to do our conversions.

Figures 14 through 17 show the measured signal on the input signal from the AD2, in yellow, and the resulting EEG output from U6 before going into the ADC, in blue. The input into the EEG was 100 mV for this test but the frequency increased from 10 Hz to 20 Hz to 60 Hz and finally 100 Hz in order of the figures. Figures 14 and 15 show our initial signal around 70 mV being amplified to around 210 mV and 143 mV, respectively, which shows the EEG allowing through the frequencies in our range from 8Hz-30Hz. Then Figure 16 and 17 also show around 70 mV on the input but have nothing showing on the blue line. This is because both of those frequencies

are outside the range of our op amps and are being filtered out by the notch and low pass filters in the circuitry.

## 2.4. *Conclusion*

The EEG subsystem is working completely as intended. The circuit was designed to filter out frequencies below 8 Hz and above 30 Hz as our focus is on the alpha and beta wave ranges when determining fatigue. We also have the two 60 Hz notch filters to help filter out extra power line interference we may experience when using the device in noisy environments. The EEG is also supposed to amplify the input signal coming in because brain waves are usually in the microvolt range, so you need our signal to be significantly amplified for accurate data processing and analysis. The EEG was also supposed to test if we can make a reliable EEG device more affordable than commercially available ones, such as the Muse EEG that was used in the machine learning and signal processing subsystem that can be purchased for around $250-$300. The EEG we designed only cost $98.16 for all the components, batteries, electrodes, etc., see Figure 18. There is also a $33 charge for ordering the PCB to be made and shipped so all out it came out to be $131.16 all out, which is around half to a third of the price of the Muse.

| Category | Part Number | Item Name | Location to | Cost per Part | Quantity | Total Cost |
|---|---|---|---|---|---|---|
| Instrumention Amp | AD8221BRZ-R7CT-ND | AD8221BRZ-R7 Instrumentation Amp | https://ww | $10.90 | 1 | $10.90 |
| Single Op Amp | ADA4610-1ARJZ-R7 | ADA4610-1ARJZ-R7 single op amp | https://ww | $2.81 | 5 | $14.05 |
| ADC | LTC2473CMS#PBF-ND | LTC2473CMS#PBF-ND ADC | https://ww | $4.55 | 1 | $4.55 |
| | | | Totals: | $18.26 | 7 | $29.50 |
| | | | | | | |
| Capacitors | 399-7848-1-ND-Cut Tape | C0603C105K9PAC7867 (1uf) | https://ww | $0.10 | 6 | $0.60 |
| | 1276-2897-1-ND - Cut Ta | CL21A106KPFNNWE (10uf) | https://ww | $0.22 | 2 | $0.44 |
| | 1276-1176-1-ND - Cut Ta | CL05B224KO5NNNC (220nf) | https://ww | $0.10 | 4 | $0.40 |
| | 478-1114-1-ND - Cut Tap | 0402YC103KAT2A (10nf) | https://ww | $0.10 | 1 | $0.10 |
| | 1276-1043-1-ND | CL05A104KA5NNNC (0.1uF) | https://ww | $0.10 | 7 | $0.70 |
| | 1276-1134-1-ND | CL10B225KP8NNNC (2.2uF) | https://ww | $0.12 | 2 | $0.24 |
| Resistors | RNCP0805FTD1K00CT-NI | RNCP0805FTD1K00 (1k) | https://ww | $0.10 | 2 | $0.20 |
| | RNCP0805FTD2K00CT-NI | RNCP0805FTD2K00 (2k) | https://ww | $0.10 | 5 | $0.50 |
| | RNCP0805FTD15K0CT-NI | RNCP0805FTD15K0 (15k) | https://ww | $0.10 | 1 | $0.10 |
| | YAG5090CT-ND | RT1206BRD075KL (5k) | https://ww | $0.65 | 2 | $1.30 |
| | A130442CT-ND | CRGP0603F150K (150k) | https://ww | $0.17 | 1 | $0.17 |
| | RMCF0805FT13K0CT-N | RMCF0805FT13K0 (13k) | https://w | $0.10 | 1 | $0.10 |
| | CRT1206-FZ-1002ELF | 652-CRT1206FZ1002ELF (10k) | https://w | $0.30 | 1 | $0.30 |
| | | | totals: | $2.26 | 35 | $5.15 |
| | | | | | | |
| Buck boost converter | 5V Step-Up/Step-Down V | Buck boost converter | https://ww | $14.95 | 1 | $14.95 |
| Dry Electrodes | FRI-2140-1E | Package of 15 Disposable/Reusable Dry E | https://ww | $34.95 | 1 | $34.95 |
| Alkaline Batteries | | Amazon Basics AA 1.5 Volt Performance A | https://ww | $5.49 | 1 | $5.49 |
| Battery Holder for 4 Batteries | | LAMPVPATH (Pack of 2) 4 AA Battery Hold | https://ww | $5.99 | 1 | $5.99 |
| 2 Output Pin Termina | 732-2028-ND | 6.91103E+11 | https://ww | $0.71 | 1 | $0.71 |
| Power 3 Input termin | 732-2027-ND | 6.91102E+11 | https://ww | $0.71 | 2 | $1.42 |
| | | | totals: | $62.80 | 7 | $63.51 |
| | | | | | | |
| Grand Total: | $98.16 | | | | | |

**Figure 27: Total Cost of EEG Components**

# 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. Five different machine learning models were created, trained, and tested with various datasets. These models are the Naive Bayes classifier, Artificial Neural Network, Kernel Support Vector Machine, Recurrent Neural Network, and Recurrent Convolutional Neural Network.
*Note: (initial models) - NB, ANN, KSVM used in 403*
*(final models) - KSVM, RNN, RCNN used in 404*

## 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 28: Simulator setups with subject 1 (Ali I.) on the left and subject 2 (Coady L.) on the right*

Dataset 0: This 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.

Dataset 1: This 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 Dataset 0. 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. This dataset was used to train the initial models.

Dataset O: 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.

Dataset 2: The final dataset used was constructed from Dataset 1. It contained all of the data from Dataset 1, more fully awake and fully drowsy data, and data closer to the transition state in near awake and near drowsy states for a total of 6 hours: 24 15-minute data samples containing 3 hours of awake and 3 hours of drowsy data. This dataset was used to train the final models.

### *3.2.2. Data Preprocessing*

The raw EEG signals from the datasets have been processed by the Signal Processor. The initial models (NB, ANN, KSVM) used the initial theta-alpha-beta signal processor where processed data was in the form of 12 features: 3 features for each of the 4 electrodes from the Muse. The final models (KSVM, RNN, RCNN) used the final user-defined-band-limits signal processor where the data is in the form of 60 features: 15 features 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.

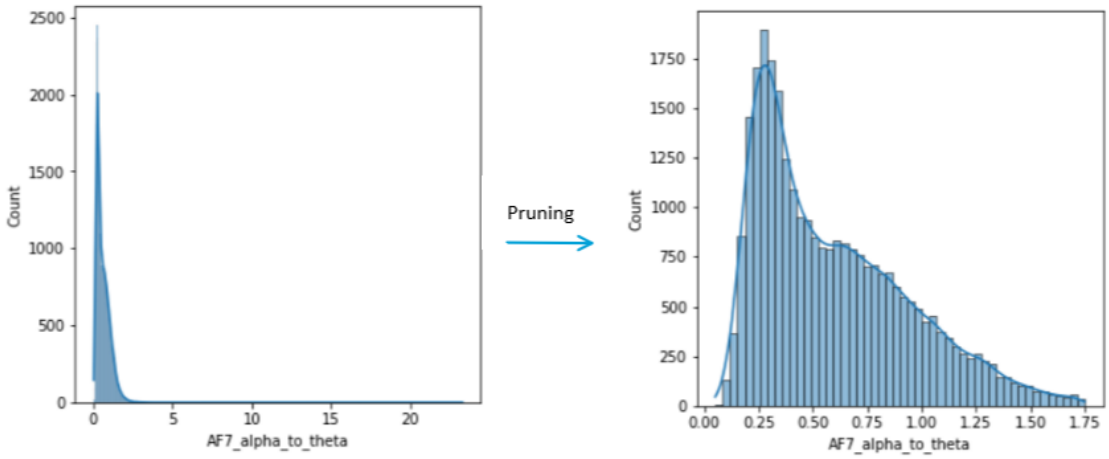



*Figure 29: Histograms of a feature (electrode AF7 alpha/theta values) before and after pruning*

Before inputting the data into the neural networks and Kernel SVM, the data needs to be scaled down. These 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.

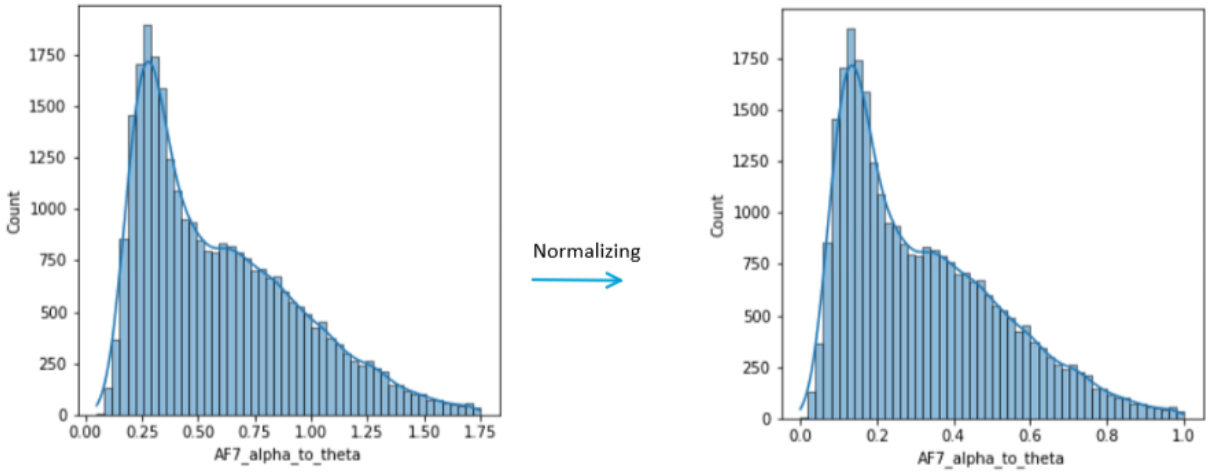



*Figure 30: Histograms of a feature (electrode AF7 alpha/theta values) before and after normalization*

### *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 (ANN)*

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.

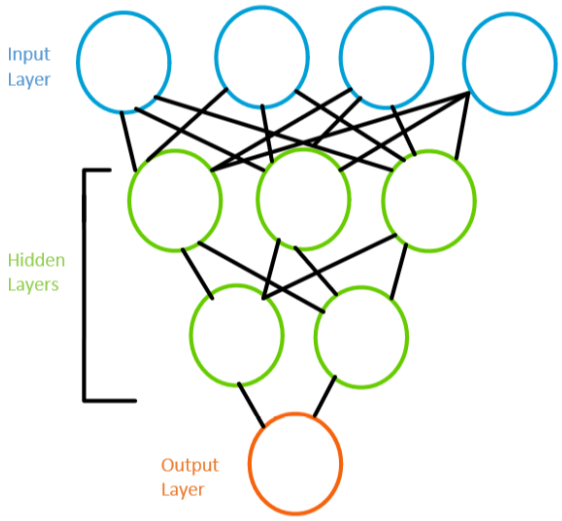



*Figure 31: Basic representation of the Neural Network's architecture*

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.

#### *3.2.3.4 Recurrent Neural Network (RNN)*

The Recurrent Neural Network model is an extension of the Artificial Neural Network (ANN). The model makes use of Long Short-Term Memory (LSTM) layers which include trainable weights that decide how much influence previously seen data has on a current output. The input to the RNN is a (12 x 60) 2-D array where 12 represents the sequence length/number of samples and 60 represents the total number of features across the 4 electrodes. The motivation for adding recurrent layers to the model came from the lack of time-series recognition in the previous models. The features represent data processed from a time-series signal and that previously seen data samples can be used to determine a new sample's output. The figures below the difference between classifying samples independently of each other and classifying samples with influence from previous samples.

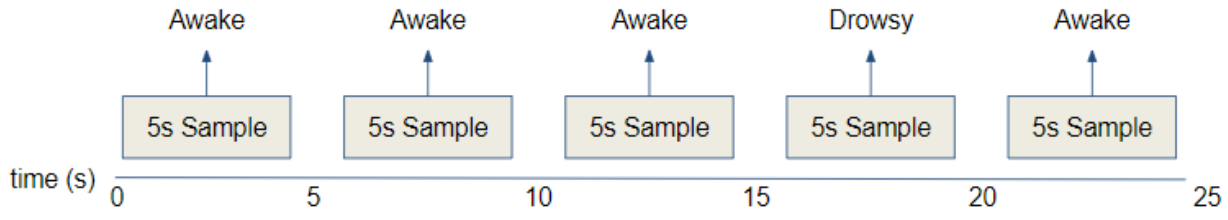



*Figure 32: Independent classification of samples*

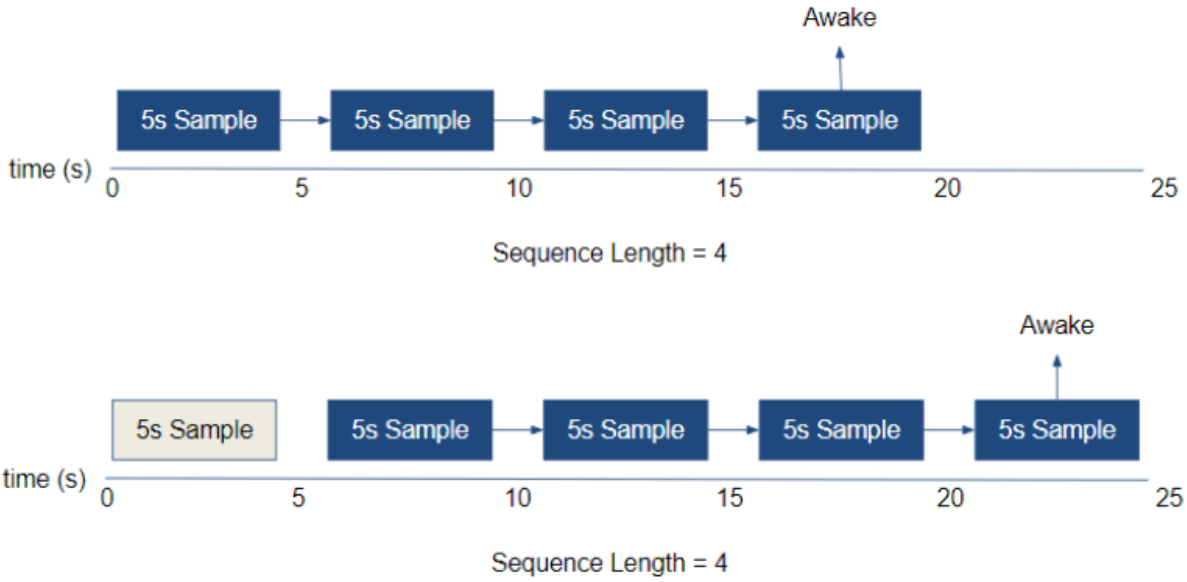



*Figure 33: Recurrency used in classifying a new sample across a sliding window of latest samples*

After experimenting with multiple different parameters, the RNN's final architecture ended up with 4 layers:

- 1 input LSTM layer with 128 units
- 1 hidden LSTM layer with 128 units
- 1 hidden dense layer with 31 neurons
- 1 output dense layer with 1 neuron

#### *3.2.3.5 Recurrent Convolutional Neural Network (RCNN)*

The Recurrent Convolutional Neural Network is an extension of the Recurrent Neural Network (RNN). The input to the RCNN is a (12 x 4 x 15 x 1) 4-D array where 12 represents the sequence length, 4 represents the number of electrodes, 15 represents the number of features per electrode, and 1 represents the feature value itself. The 60 total features are formatted in a 4x15x1 array because convolution operations are performed across the feature-set in order to recognize spatial dependencies between the four electrodes. The motivation for adding convolution layers to the model came from the attempt to better generalize across new data by discerning how features across the electrodes relate to each other. Convolution is normally used in classifying images because features can be extracted by looking at the spatial dependencies between pixel values across an image. The same logic is applied here in extracting new features based on spatial dependencies between the electrodes and their features by formatting the feature set as an "image".

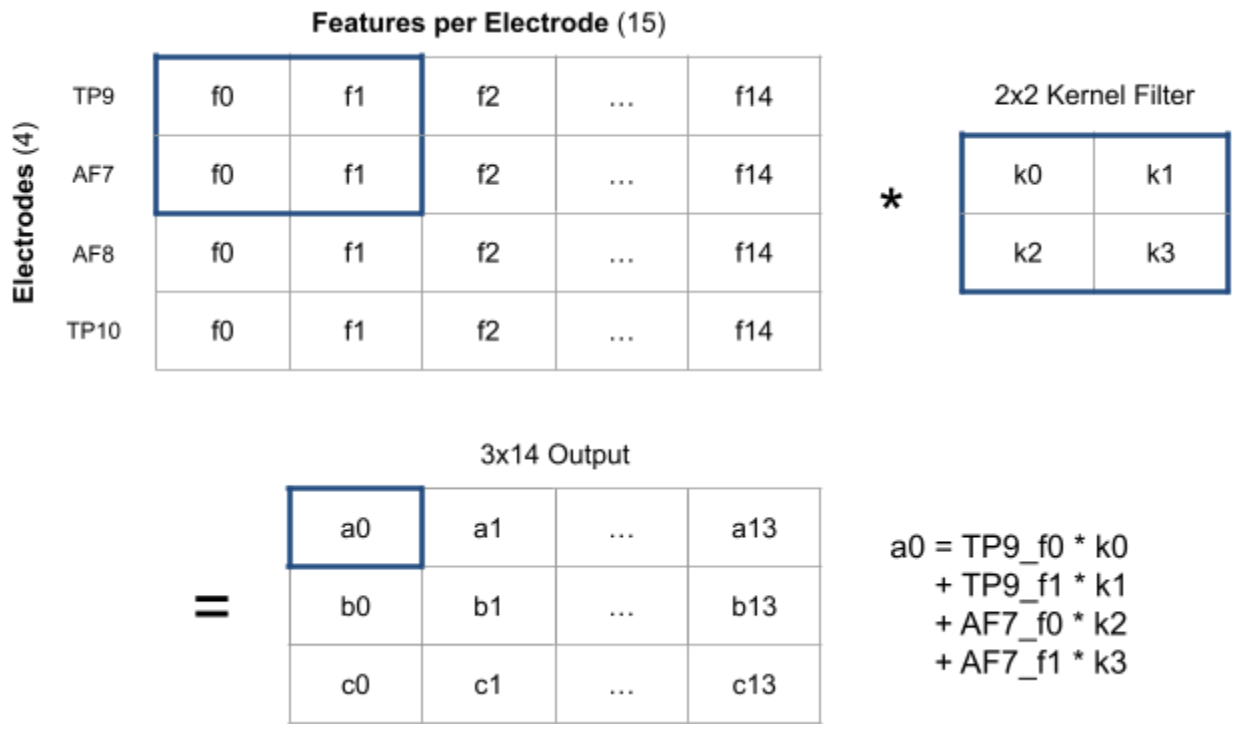



*Figure 34: Convolution performed between the first 2x2 TP9-AF7 feature block and the 2x2 kernel filter*

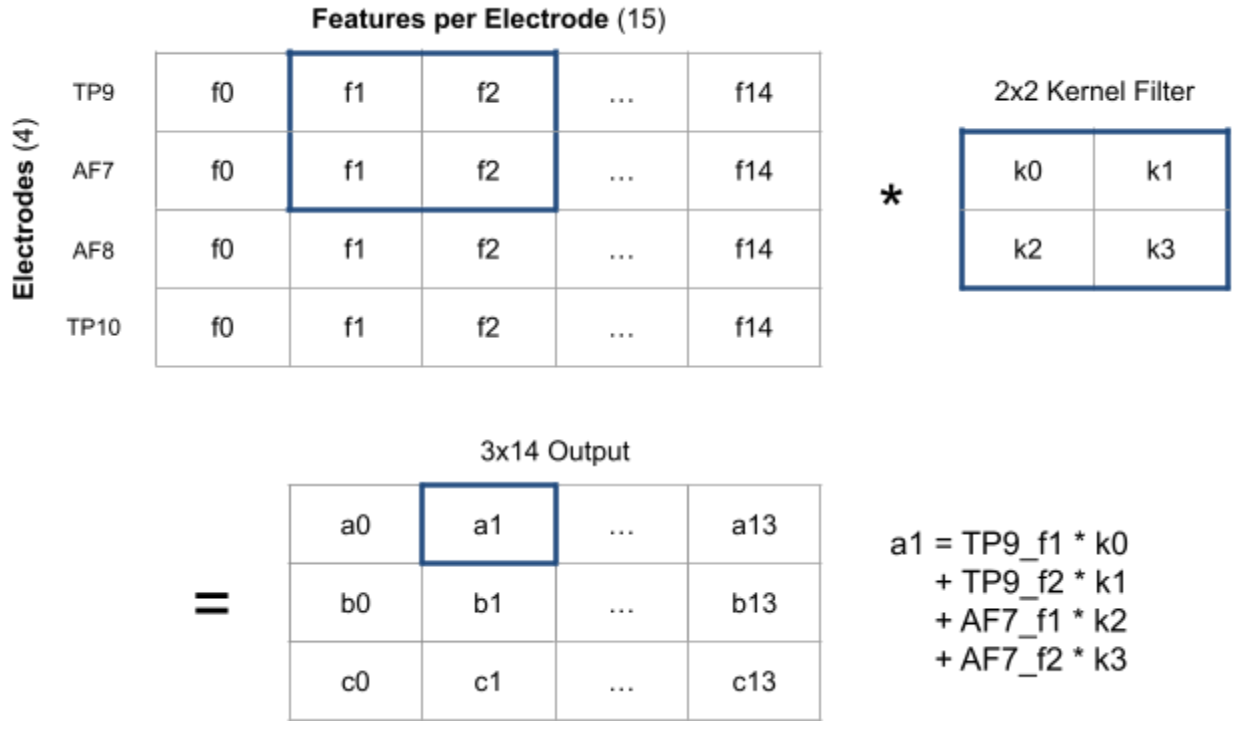



*Figure 35: Convolution performed between the second 2x2 TP9-AF7 feature block and the 2x2 kernel filter*

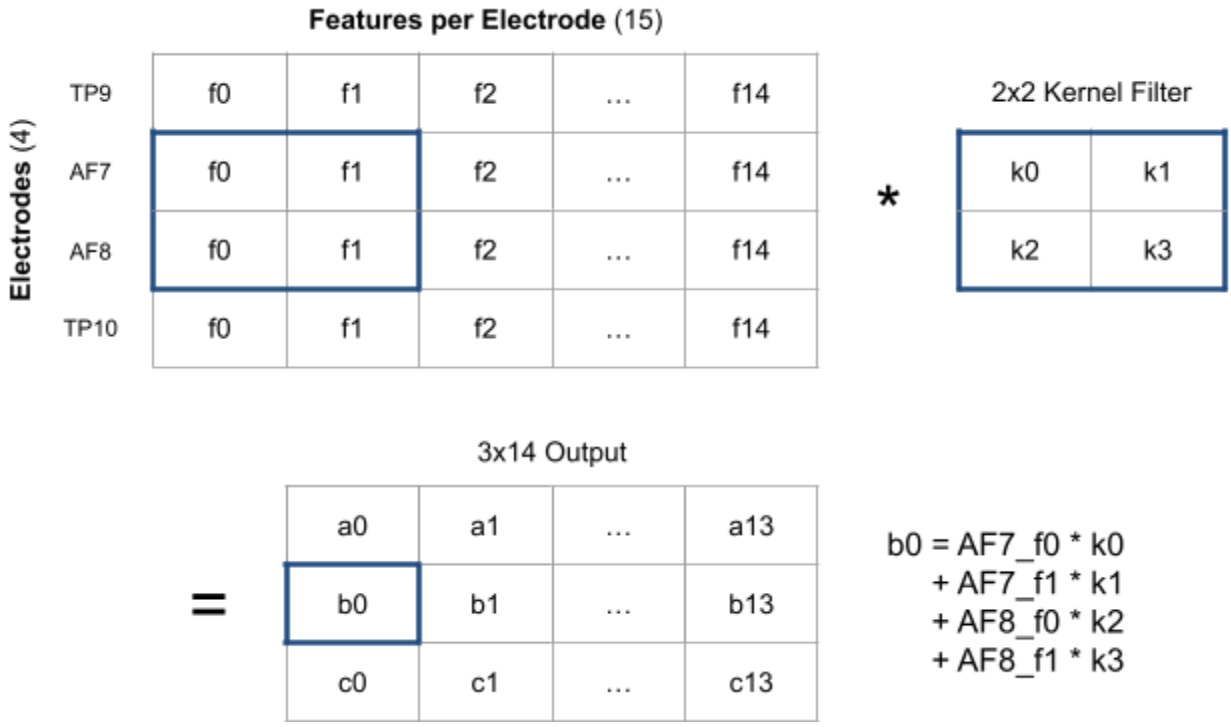



*Figure 36: Convolution performed between the first 2x2 Af7-Af8 feature block and the 2x2 kernel filter*

A 4x15x1 feature sample is run through two convolution layers. Both layers include a 2x2 kernel filter that convolutes over the feature array. After the first convolution, 64 3x14x1 outputs are generated from 64 different kernel filters. These outputs are then sent to the second convolution layer where 32 2x13x1 outputs are generated from 32 new kernel filters. These outputs are then sent to a max pooling layer which converts a 2x13x1 array into a 1x6x1 array which is then flattened into a 1x6 array. 192 (6 element array * 32 filters) new features are generated based on the spatial dependencies between the four electrodes. This feature set is then sent to the recurrent layers.

After experimenting with multiple different parameters, the RCNN's final architecture ended up with 7 layers:

- 1 time distributed input 2D convolution layer with 64 filters
- 1 time distributed hidden 2D convolution layer with 32 filters
- 1 time distributed hidden 2D max pooling layer
- 2 hidden LSTM layer with 128 units
- 1 hidden dense layer with 31 neurons
- 1 output dense layer with 1 neuron

The convolutional layers are embedded in time distributed layers because the 12 samples in the sequence each need to be run through the convolutional layers.

## 3.3. *Validation*

The Machine Learning models were validated on two main metrics: accuracy of the model on testing data and speed of prediction. More results during real-time classification are presented in the accuracy and transition validation results are provided in the System Validation document under the Software System Validation section.

### *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 networks, 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. The limitation from using this testing data is that it contains data from the same *dataset* that includes the training data.

#### *3.2.1.1. Initial Models*

| Accuracies (%) | Dataset | | |
|---|---|---|---|
| Model | Dataset 0 | Dataset 1 | Dataset O |
| Naive Bayes | 71.7 | 74.6 | 72.9 |
| ANN | 78.3 | 83.5 | 97.1 |
| Kernel SVM | 83.5 | 85 | 97.3 |

*Table 1: Testing accuracies of the initial 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 ANN 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.

Neural Network:

```
[[2164  645]
 [ 401 3181]]
[0.80535914 0.8588013 ]
Total accuracy:  0.8353121665723426
```

Kernel SVM:

```
[[2245  618]
 [ 333 3195]]
[0.82521595 0.87045362]
Total accuracy:  0.8501883324862117
```

*Figure 15: Confusion matrices of Neural Network and Kernel SVM over Dataset 1*

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.

#### *3.2.1.2. Final Models*

| Accuracies (%) | Dataset | |
|---|---|---|
| Model | Dataset 1 | Dataset 2 |
| Kernel SVM | 92.4 | 88.2 |
| RNN | 97.7 | 95.4 |
| RCNN | 98.9 | 96.1 |

*Table 2: Testing accuracies of the final ML models over relevant datasets*

The RCNN performed the best out of all the final models on both datasets. Each model also performed slightly better on Dataset 1 than on Dataset 2. Dataset 1 only contains fully awake and fully drowsy data whereas Dataset 2 contains some data closer to

the transition state (near awake and near drowsy) which is why the models perform slightly worse. However, Dataset 2 is the more relevant dataset since the models will be tested on transition state data during real-time classification.

The Kernel SVM performed the worst because it does not take into account previous data samples in order to create a new output on the current data sample. This is the advantage that both the RNN and RCNN have. The RCNN performed slightly better than the RNN because of the convolutional layers that look at spatial dependencies between the electrodes, allowing for greater generalization. After doing real-time classification on both the RNN and RCNN, the RCNN also made fewer misclassifications. The RCNN is chosen as the final model used in the Software System based on these results.

### *3.3.2. Speed*

Speed was used as another metric for validation because classification of EEG signals will 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 processed by the Signal Processor.

#### *3.2.2.1. Initial Models*

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) |
|---|---|
| ANN | 0.005 |
| Kernel SVM | 0.05 |

*Table 3: Prediction speed of the ML models over 5 seconds of new data*

Both the ANN 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 ANN was roughly 10 times faster than the SVM.

#### *3.2.2.2. Final Models*

The final ML models were trained with 5-second samples. Therefore, the target prediction speed was established as predicting in < 2.5 seconds over a 5 second chunk of data. This was to allow a target computation speed of < 2.5 seconds for the Signal Processor.

| Model | Speed (s) |
|---|---|
| Kernel SVM | 0.05 |
| RNN | 0.11 |
| RCNN | 0.12 |

*Table 4: Prediction speed of the final ML models over 5 seconds of new data*

All three models performed well under the limit of 2.5 seconds for prediction of a single 5-second sample of data.

## 3.4. *Conclusion*

After experimenting with multiple machine learning models, the Recurrent Convolutional Neural Network ended up being the most accurate across testing data and made the fewest misclassifications during initial real-time classification testing. Much of the development of the ML models came down to trial-and-error. Although there are certain recommendations for certain parameters, each machine learning application is different and requires an iterative development process where multiple different kinds of models are thoroughly trained and tested on meaningful data.

# Signal Processing Subsystem

## 3.5. *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. The code was generalized to handle any number of bands with user defined limits in 404, and a mode was added to normalize any number of bands to the highest band. Additionally, new methods of data augmentation were added to accommodate the transition to sequential mode for the ML model input. These include both randomized data augmentation and offset. Finally, live analysis was modified to output sequential data to the ML model.

## 3.6. *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.

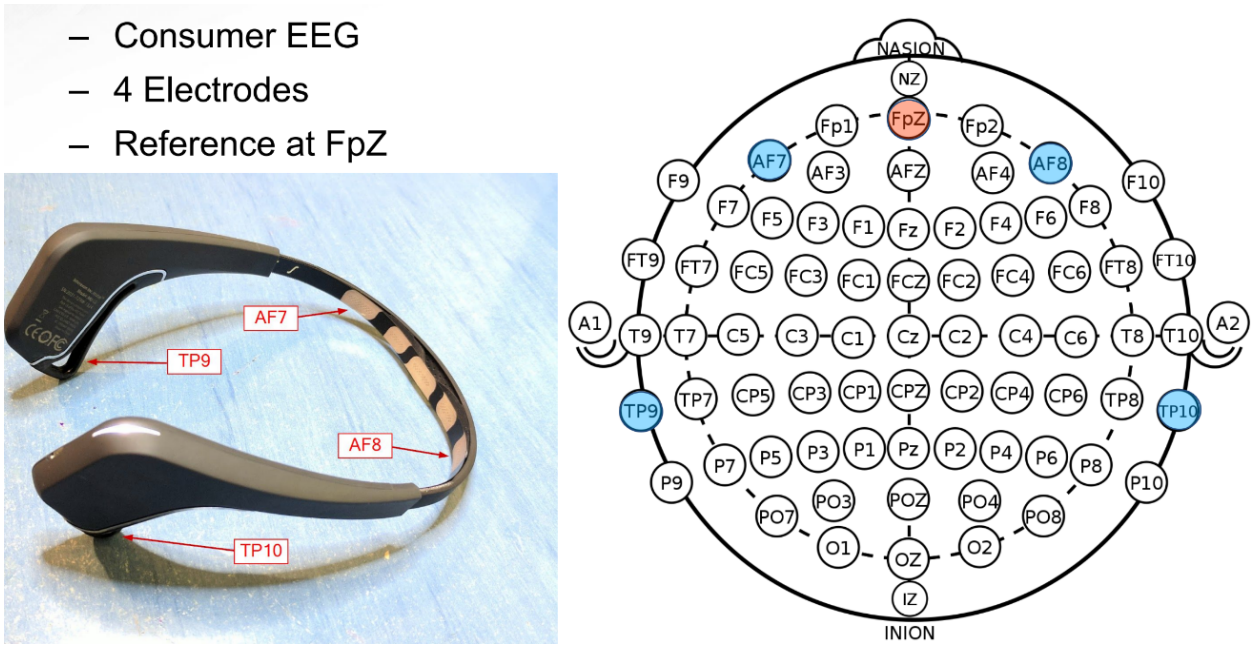



*Figure 38: Muse Electrode Placement on International 10-40 Standard*

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 in the original mode, 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 dropped.

The first iteration of the recording analysis calculated the alpha/beta ratio and second calculated the individual theta, alpha, and beta power values. The third iteration calculates alpha/beta, theta/beta, and alpha/theta ratios, and it can also handle any number of electrodes, such that online datasets could be used for training.

A few changes were made to the original code in 404. First, the calculation of ratio features was generalized to handle any number of bands with band limits of the user's choosing. The ratio features were then calculated combinationally, so 6 bands produced ${}_6C_2$ = 15 features per electrode. This addition gave the largest increase in accuracy from the signal processing system alone, with consistent bumps of 9-10% by increasing from 3 to 6 bands. Additionally, a mode was added to take any number of specified bands and normalize all of them to the highest band to produce N-1 features. The highest band was chosen because it has little activity and is relatively stable in our application. In limited testing, this showed no change in the testing accuracy, but it was not tested on new real-time datasets due to time constraints. Processing data is by far the most time-intensive part of training a new model, and each time parameters are changed, it must be done again. Good results were achieved using N ratio bands, so that mode was used while leaving the second one as an option for future testing.

Although having more bands should always give more information about the state, there is a practical limit when coupled with the machine learning model, especially with limited training data. Increasing the number of bands causes a large increase in the number of features; going from 6 to 7 bands adds ${}_7C_2$ - ${}_6C_2$ = 6 features to each electrode, taking the total feature number from 60 to 84. A larger number of features requires significantly more data to train a ML model well, so this strategy cannot be applied without restraint. As a compromise, the band density (shown below) is higher in areas where lots of variation and information is expected in the signal, namely 4-12 Hz. More bands could be added in future iterations if sufficient training data is collected, and the accuracy of the model would be improved.

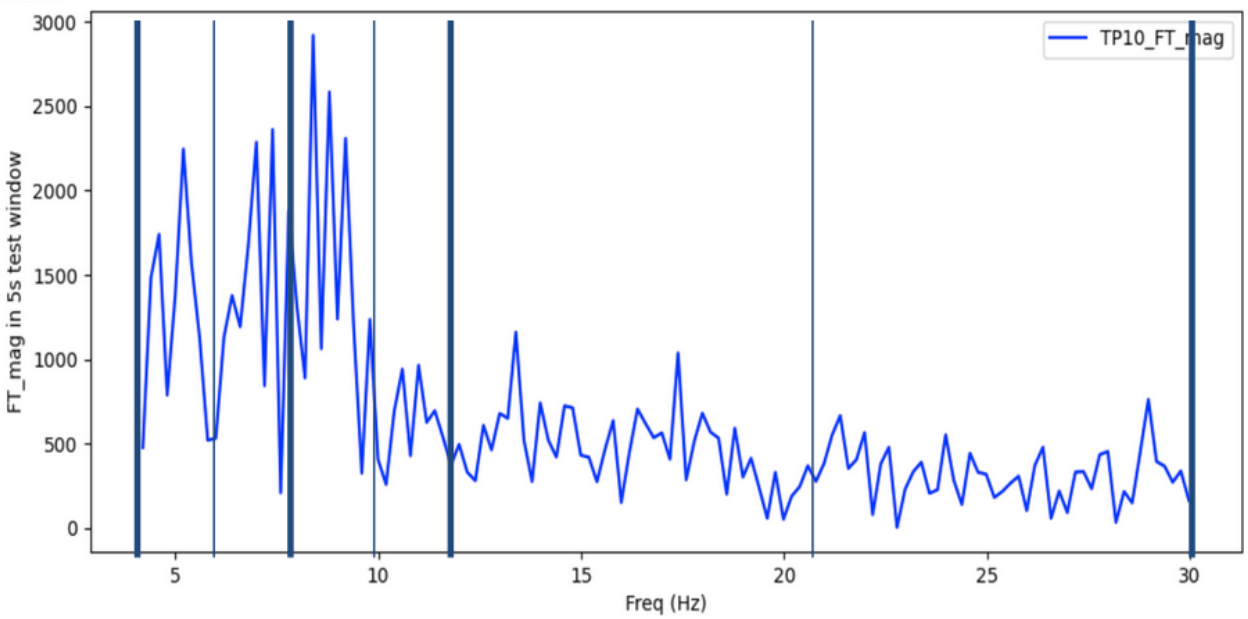



*Figure 39: Band Distribution*

Switching to a sequential model requires a different data augmentation strategy. Since the outputs in real time operation must be sequences with adjacent windows, the training data must also be in this format, so overlapped windows will not work. Initially, randomized data augmentation was used to create sequences with transitions between awake and drowsy states, and that script is included. It works by taking processed 5 second chunks, assigning the number of transitions randomly, and placing the transition partitions randomly. In initial testing, the accuracy of the ML model was not improved, and this is likely because there is just not enough data to extend it in this manner. If there were more data points, there would be a lower probability of having points repeated in the augmented sequences.

The selected augmentation strategy uses offset to generate new data sequences. By offsetting every window in a sample by less than one window size as shown below, unique new data sequences can be generated. This is important when working with lower data volume. Even though there can be dependencies, the uniqueness of the new data makes this a stronger method than randomized augmentation, which can repeat the same point several times.

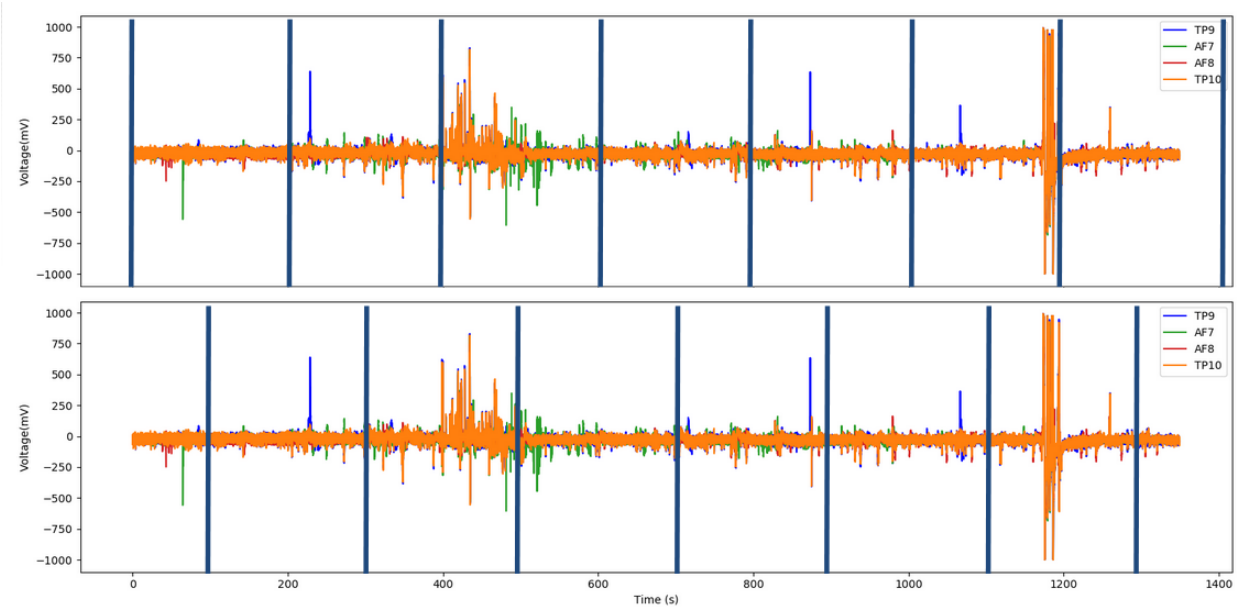



*Figure 40: Offset Visualization*

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.

## 3.7. *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.

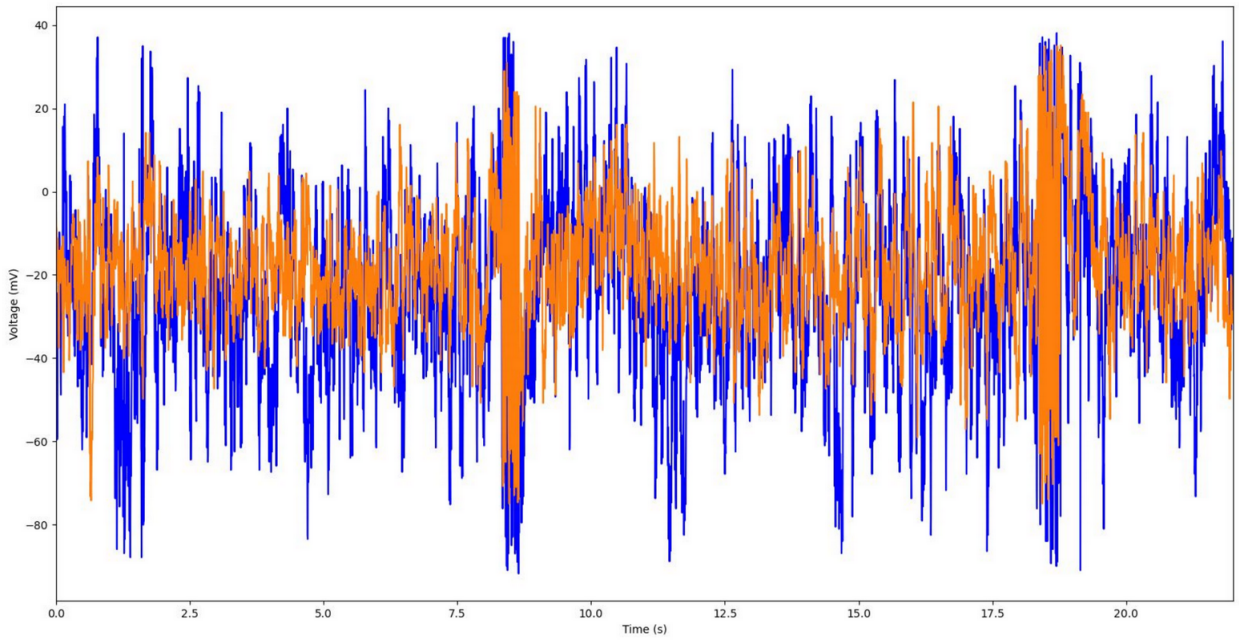



*Figure 41: Raw EEG Signal from Eye Test*

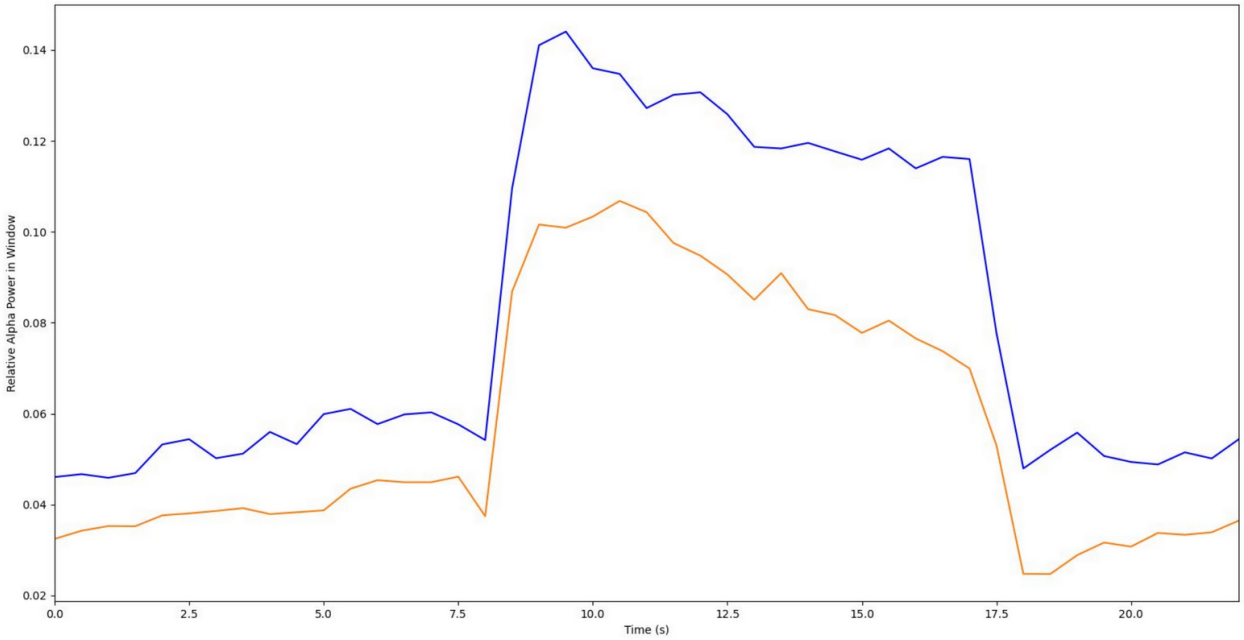



*Figure 42: alpha/beta Power Ratio from Eye Test*

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.

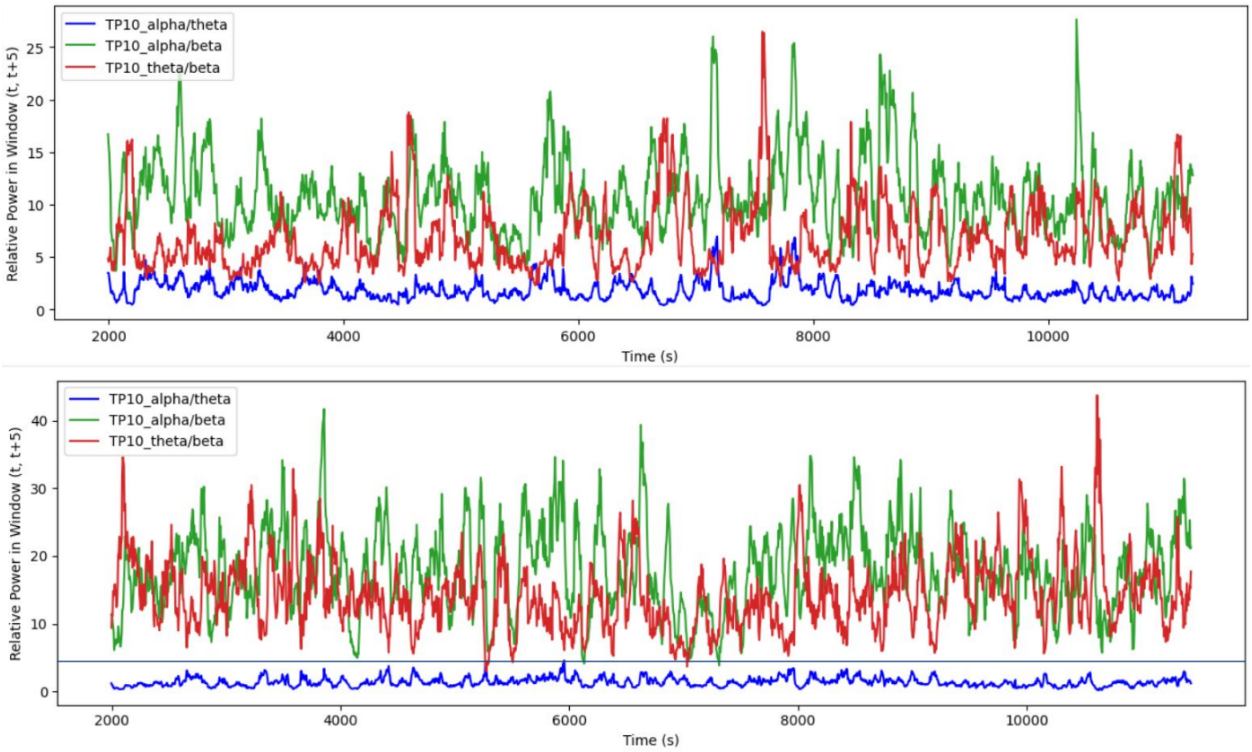



*Figure 43: Awake (Upper) vs Drowsy (Lower) in Mode 3 of the Analysis (TP10 electrode)*

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.

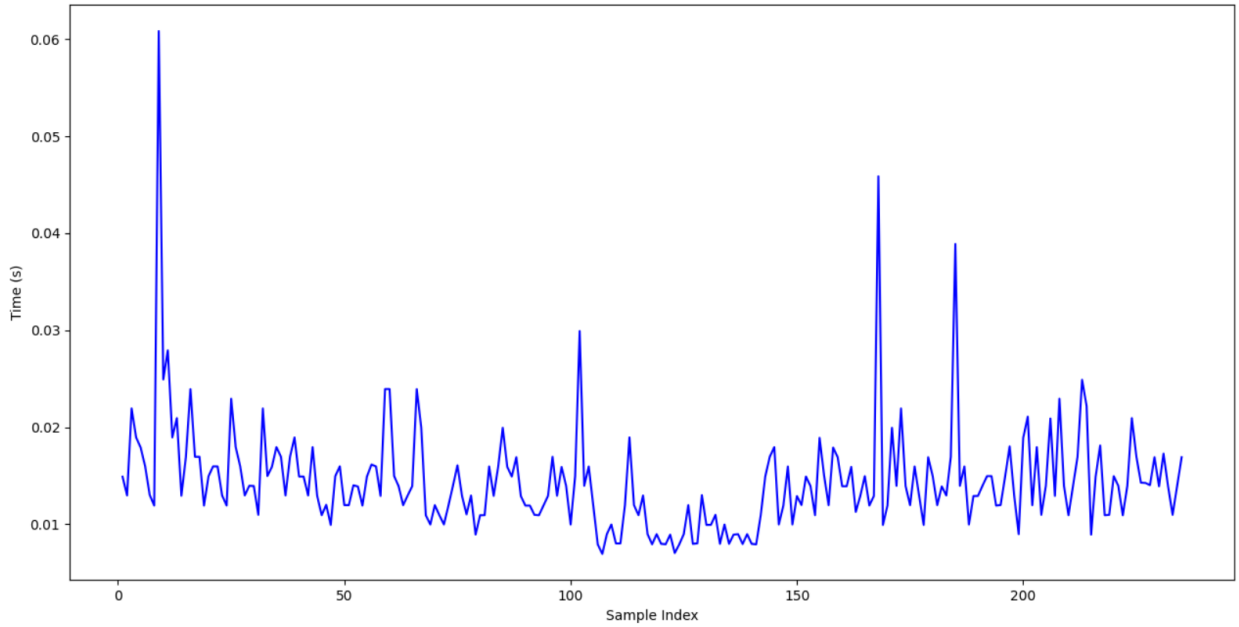



*Figure 44: Computation Times on 5s Intervals at the Muse Frequency (20 minute test)*

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.

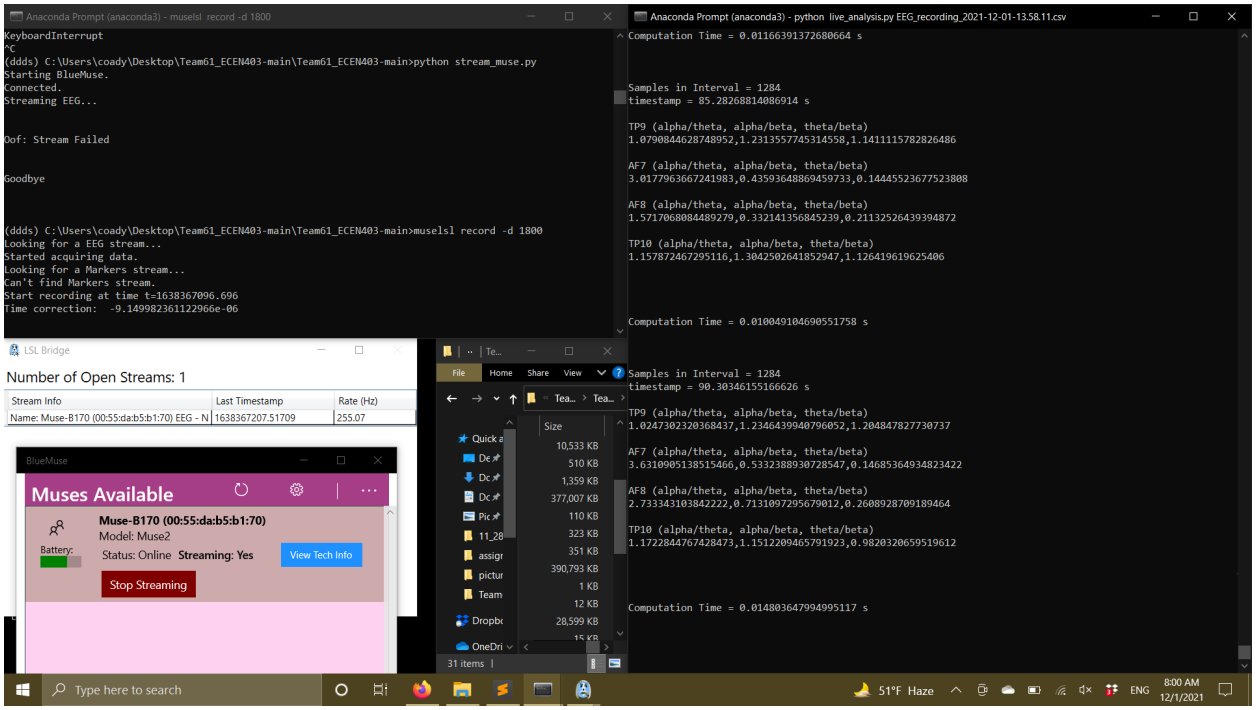



*Figure 45: Live Analysis Running on the Muse Stream (5s Intervals, 256 Hz)*

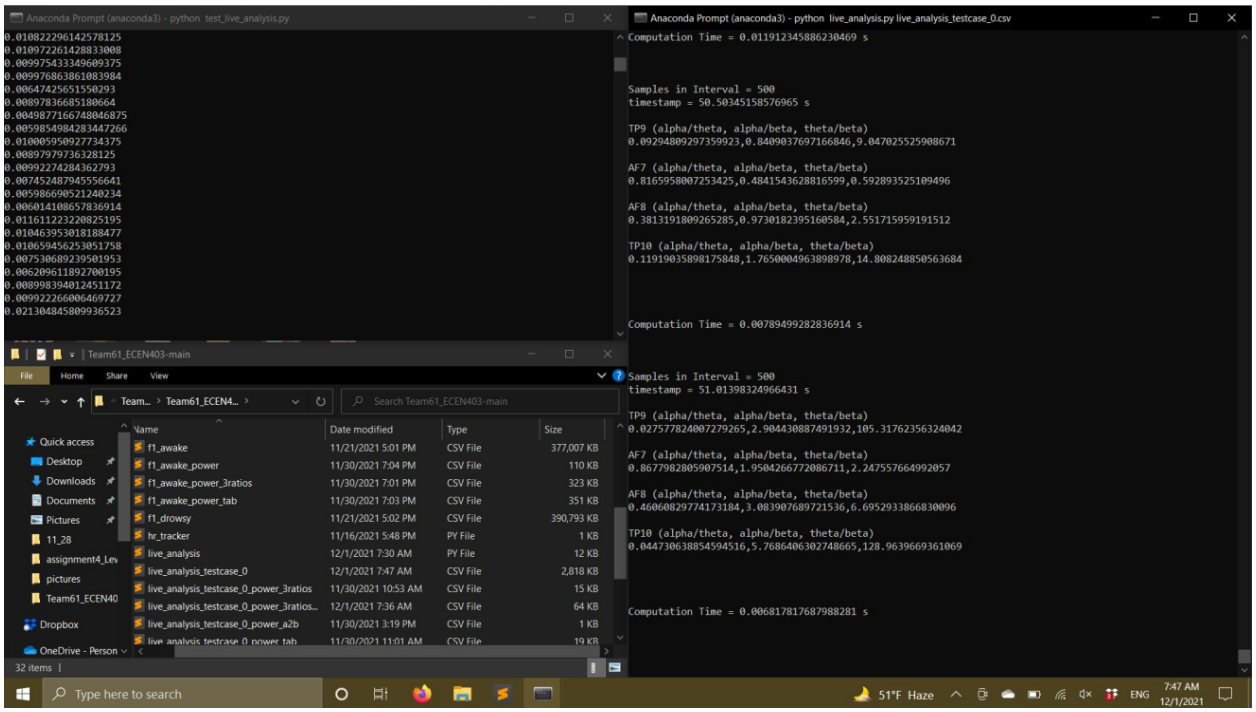



*Figure 46: Live Analysis Running on the Testbench (0.5s Intervals, 1000 Hz)*

## 3.8. *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 have been very strong. Full software integration has been achieved, and the live analysis is able to output sequential data quickly enough to perform real-time fatigue state classification with the Muse device. The main area where improvements could be made is the band density. If more training data is acquired, adding more bands in critical regions will yield improved results.

# Driver Drowsiness Detection System

Dakota Mouton, Ali Imran, Coady Lewis

# System Validation Report

REVISION – Final Draft
30 April 2022

# System Validation 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 |
|---|---|---|---|---|
| - | 4/30/2022 | Entire Team | | Draft Release |

# 1. Introduction

In regard to integration, our sponsor's requests were as follows.

The software system needed to accurately classify fully awake and fully drowsy data from the subjects on which it was trained. It needed to function quickly enough to classify the subjects' states from a live feed of Muse data. Finally, it needed to show the transition process from fully awake to fully drowsy states on each subject. The included videos and graphs below were the validation metrics requested, and the software was able to meet these expectations.

The hardware was expected to filter out noise outside of the 8-30 Hz band of interest, as well as output and record the amplified signal from the board input. The hardware was able to do this for waveform input.

# 2. Hardware System Validation

## 2.1. *Introduction*

The integrated hardware system consists of the custom EEG PCB and an Arduino Uno R3 as the MCU to interface the EEG output signals to the computer for processing and live graphing of the signal. The EEG device is meant to take in a differential signal from two electrodes attached to the head of the user, amplify the voltage to a maximum of 1.25 V through the IC's and filter out frequencies outside of our target range, then send that raw data into the analog to digital converter, or ADC, to then be converted to digital signal that the MCU will take in via I2C communication and process that data into a live grapher.

## 2.2. *Validation Components*

### 2.2.1. The MCU (Arduino Uno R3)

The Arduino Uno R3 is used to connect the EEG circuitry to my computer in order to run a visual live graph that plots the converted voltage from the ADC and also records the values into an excel sheet for further processing or analysis. The MCU is powered via USB through the computer and shares a reference ground with the rest of the circuit for accurate measurements. The ADC on the EEG is also connected via I2C serial communication utilizing the SCL and SDA lines for matching clock timings and data retrieval.

### 2.2.2. The Live Grapher

The live grapher python code does function and plots the live signal from the ADC, but the rate at which the conversion is happening is slower than I was hoping for. I was not able to get the code to work at a faster rate in the time between final demo and the writing of this report so the graph will be more stretched out than usual but is still plotting the result.

## 2.3. Validation Metrics

### 2.3.1. Plots the converted voltage data from ADC

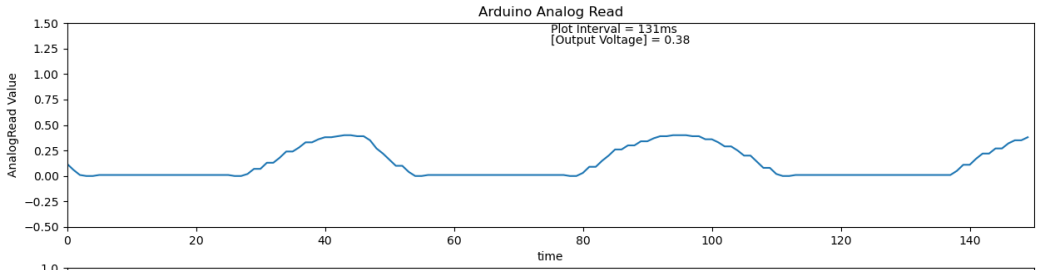



**Figure 47: 20Hz, 50mV Output Graph**

The result ended up getting up to around 0.45 V as the output measurement which is pretty close for this low of an input voltage.

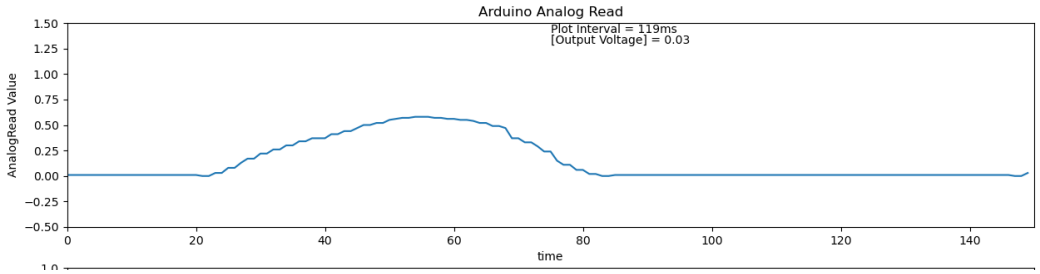



**Figure 48: 10Hz, 50mV Output Graph**

The result on this graph is more stretched out as it is at a lower frequency and my code makes it run around 1/10th the speed it should be.

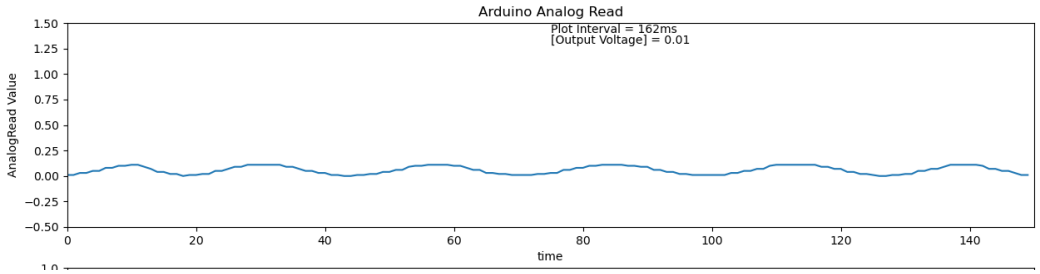



**Figure 49: 50 Hz, 100mV Output Graph**

The result on this graph is staying around 0 V since the frequency is being filtered out by the EEG, therefore showing the live grapher responding to the filtering as well.

## 2.4. Conclusion

The hardware system as a whole functions as intended. The EEG device is amplifying the output voltage reading and filtering out the unwanted frequencies outside the range of 8Hz-30Hz and the MCU is receiving the data from the ADC and plotting that data into the live plotter for better customization and accuracy than the arduino ide plotter and also records the data into an excel sheet for further analysis or record keeping. There was an issue somewhere in my arduino code that was causing the signal conversion process to run much slower than it should but if that code were to be fixed in a future iteration then it should all be ready to go for the data collection and plotter.

# 3. Software System Validation

## 3.1. Introduction

The integrated software system included the Signal Processor subsystem and the Machine Learning Model subsystem. The goal of the software system was to collect data from the Muse 2 EEG device, send the data to the signal processor, send the processed data to the already trained machine learning model, and display the classification results, all in real time. There were three main metrics used in validating this system: real-time accuracy on fully awake and fully drowsy signals, displaying the transition from awake to drowsy classifications over a long period of time, and speed of the computation and classification. The RCNN was the chosen ML model as it generalized the best across new, unseen data.

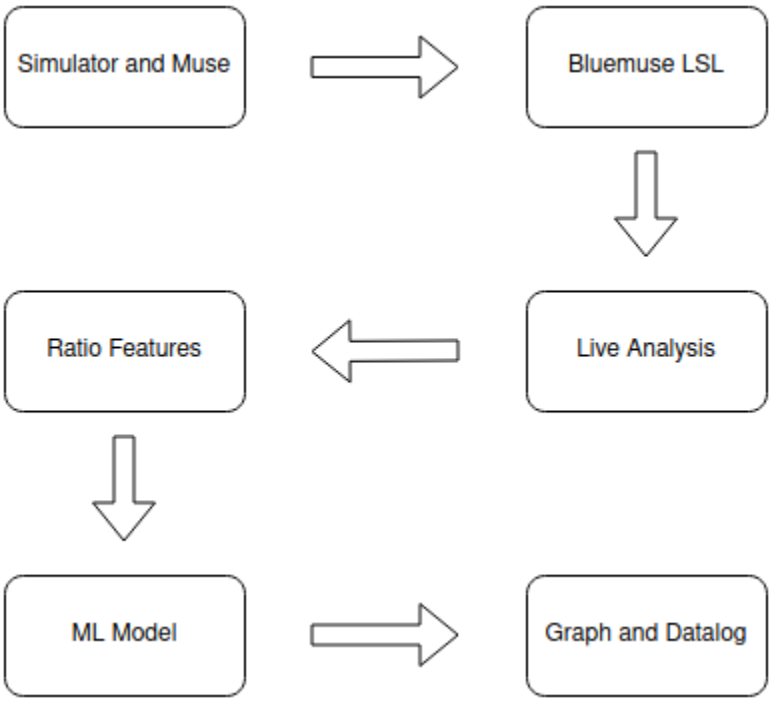



*Figure 50: Driver Drowsiness Detection Software Block Diagram*

## 3.2. Validation Components

This section describes the components and systems created to support validation of the Software System. These components include the programs for live analysis and the live analysis graph.

### 3.2.1. Live Analysis Script

The live analysis script was modified from the version at the end of 403 to include more helper functions for modularity and clarity. Support for sequential data output to the

model was also added, as well as support to run any of the model types used during 403/404.

### *3.2.2. Live Analysis Graph*

The live analysis graph was created to visualize the real-time classification outputs. The graph was a sliding window that would be updated whenever a new classification output was made. A new script was created to instantiate two processes connected by a shared channel. The shared channel was a named pipe which allows for interprocess communication. One process ran the live analysis script and wrote the classification outputs over the pipe. One process ran the live analysis graph which read the classification outputs from the pipe.

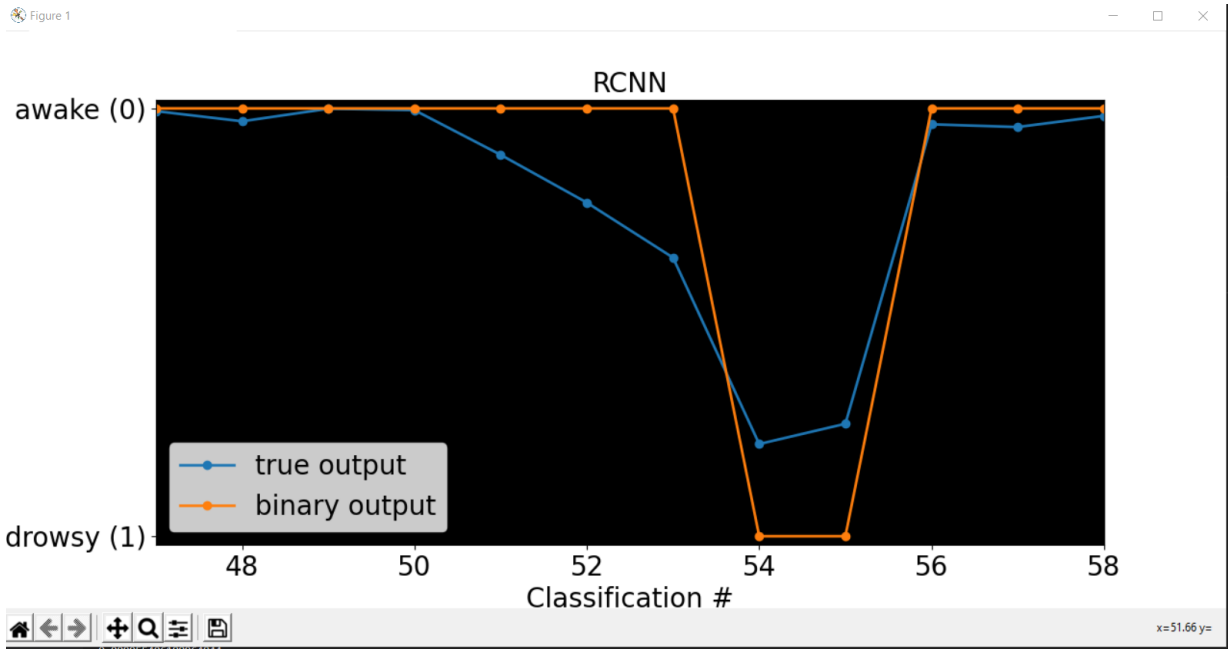



*Figure 51: Snapshot of Live Analysis Graph over a sample of Ali's awake data classified by RCNN model*

## *3.3. Validation Metrics*

### *3.3.1. Accuracy*

The major validation metric for the Software System was achieving an accuracy of at least 90% over fully awake and fully drowsy data. The subjects that trained the model with their data (Ali I. and Coady L.) each collected data over the simulator during fully drowsy and fully awake sessions. Videos uploaded to YouTube showcase the real-time classification of the model on the subjects during awake and drowsy states.
YouTube Links:

- Timelapse: https://www.youtube.com/watch?v=pWFG5qb1vY0
- Coady Drowsy: https://youtu.be/5ImO0vi3z6A
- Coady Awake: https://youtu.be/bnNPSJj9xhM
- Ali Drowsy: https://youtu.be/vPEunQcCHIk

- Ali Awake: https://youtu.be/8iqVuxgb4KY

The “Timelapse” video is a 32x sped up compilation of the four other videos.

The accuracies shown below were taken from the real-time classification of 400 fully awake samples and 400 fully drowsy samples by the RCNN.

- Total: 93.6%
- Fully Awake: 92.3%
- Fully Drowsy: 95%
- False Positive: 7.7%
- False Negative: 5%

The validation accuracy of 90% was surpassed as the total accuracy was 93.6%. The false positive rate (true awake being mislabeled as drowsy) was higher than the false negative rate (true drowsy being mislabeled as awake). When misclassification does occur, it would be preferable that an awake sample is mislabeled as drowsy because a drowsy sample being mislabeled as awake could potentially be dangerous in a system that needs to alert a driver of their fatigue state.

### *3.3.2. Transition Detection*

Transition detection refers to the ability of the Software System to showcase the transition from classifying awake to drowsy signals as a subject transitions from an awake state to a drowsy state over a long period of time. The subjects that trained the model with their data (Ali I. and Coady L.) each engaged in collecting “long session” data. Long session data refers to data collected in 10-minute chunks every hour over several hours when the subject transitions from an awake to drowsy state. The results of these long sessions are displayed in the graphs below.

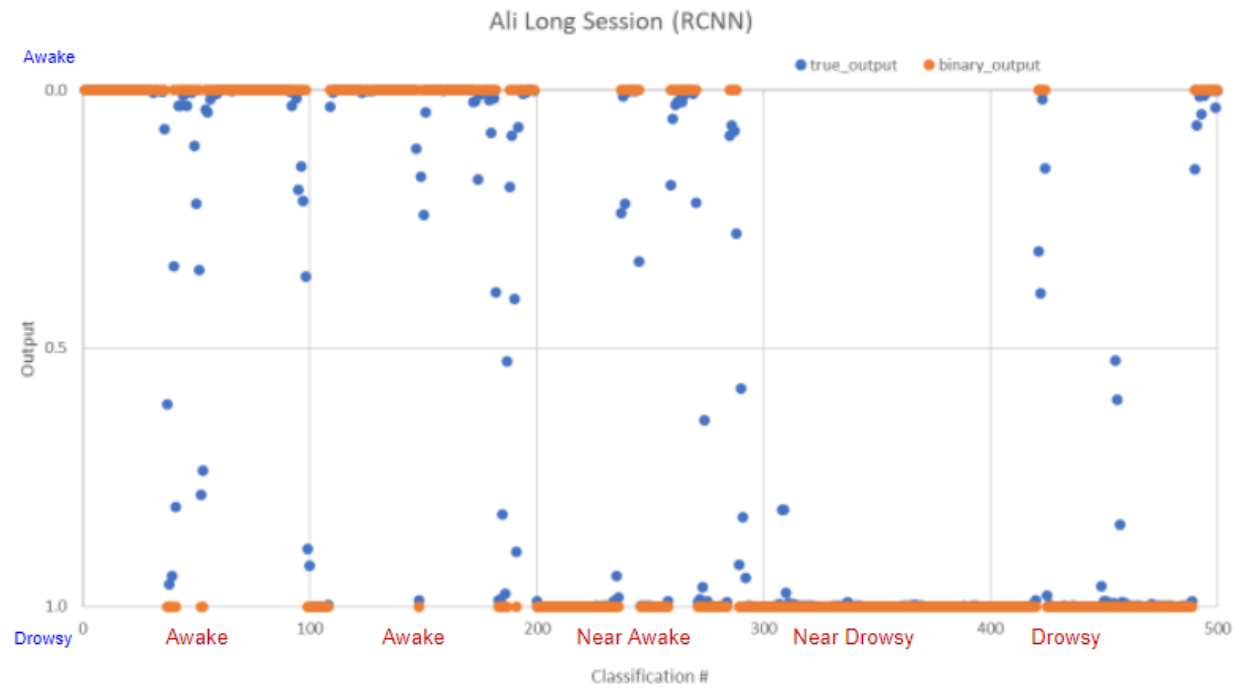



*Figure 52: RCNN classification of Ali's long session data*

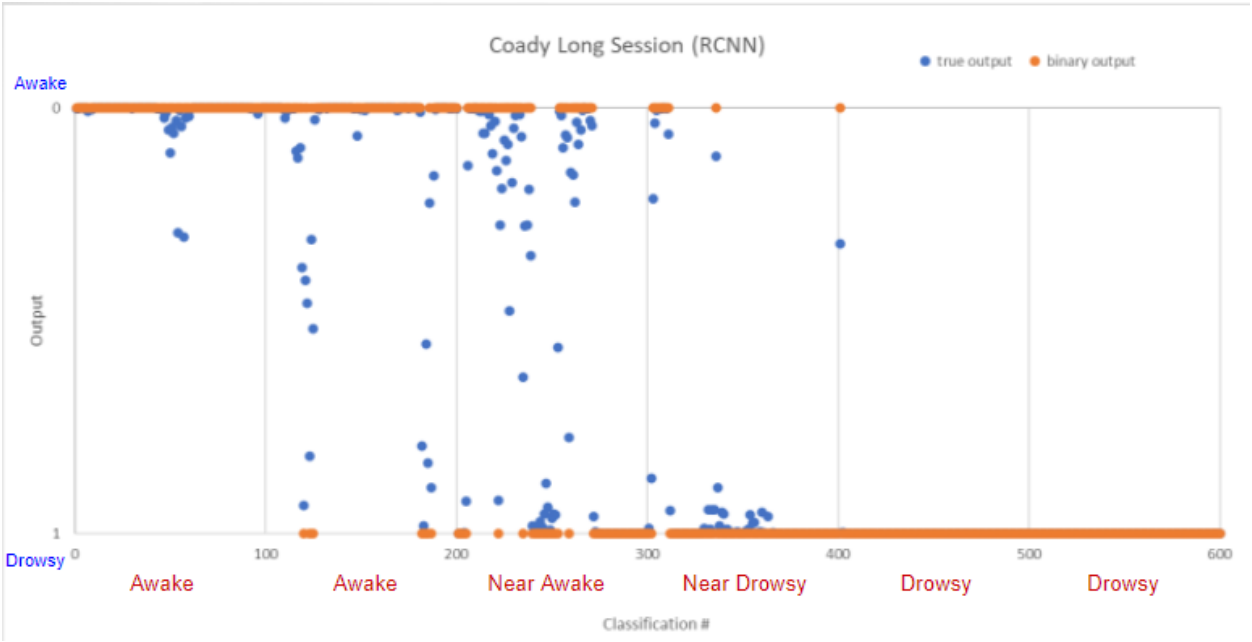



*Figure 53: RCNN classification of Coady's long session data*

Each of the long session graphs show two plots: one is the "true output" from the RCNN which is a continuous value between 0 and 1, one is the "binary output" which takes the true output and rounds it up to 1 (drowsy) or down to 0 (awake). In both cases, the model is able to display the transition from fully awake to fully drowsy. The fully awake

states (Awake) and fully drowsy states (Drowsy) are accurately classified. The inner states (Near Awake and Near Drowsy) are considered the “transition state” where someone starts to approach a drowsy state. Oscillation occurs between Awake and Drowsy classifications during the Near Awake state while the Near Drowsy state is classified mostly as Drowsy.

The system was also able to respond to sudden changes in awake states. During one of Ali’s drowsy sessions on the simulator, he ends up driving into a short post that gets stuck in front of the truck for roughly 20 seconds. Ali is observed to be in a much more alert state than before, and the live analysis graph captures the model classification outputs going from drowsy to awake during that alert state and then back to drowsy. The YouTube video linked below shows the scenario (post is hit roughly 40 seconds in).
Response to Stimulus: https://youtu.be/Y3JlhLka4Ks

### *3.3.3. Speed*

The last metric of validation was ensuring that the Software System could process and classify a data sample in less than the data sample’s elapsed time. Since 5-second data samples were processed and used to train the model, the system had to process and classify a new 5-second sample in less than 5 seconds. If more than 5 seconds was used to classify the output, the output classifications would lag behind the newly collected data sample.

Across 200 5-second classifications, the average total computation time (including time spent processing the 5-second signal data and time spent classifying the sample with the RCNN) was **0.1813 seconds** with a standard deviation of 0.01849 seconds. The video links listed above (in sections 3.3.1 and 3.3.2) also show the model outputs being classified in less than 5 seconds during real-time classification.

## *3.4. Conclusion*

The system was able to fulfill current expectations for the project. Future challenges on the software side include sample labeling, data volume, subject count, and simulation realism. When labeling datasets, we had to use batch labeling and decide whether we were fully awake or fully drowsy. Transition labeling was impossible as it’s difficult to classify intermediate states objectively. Also, there are times in a full sample where you may zone out, or become extra focused, and the state flips. These conditions are not labeled accurately in batch mode, so this impacts accuracy of the trained model. Changing this would require using other metrics, such as vitals, to label a subject’s state every 5 seconds for each individual sample, all without disturbing the subject. This was beyond our capabilities, but it is a potential improvement method. Next, data volume and subject count would allow for a more generalized and better trained model, and we could use higher band density in the signal processor for better accuracy. Finally, the lack of vehicle NVH on the simulator led to awake states being forced to drowsy in some cases. It is difficult to stay

focused without the real stimulation of a vehicle. The project's results are promising, and there are several legitimate pathways for improvement in the future.

# APPENDIX

GitHub Link: https://github.tamu.edu/jaxter2/DDDS_ECEN404

YouTube Links:

- Timelapse: https://www.youtube.com/watch?v=pWFG5qb1vY0
- Coady Drowsy: https://youtu.be/5ImO0vi3z6A
- Coady Awake: https://youtu.be/bnNPSJj9xhM
- Ali Drowsy: https://youtu.be/vPEunQcCHIk
- Ali Awake: https://youtu.be/8iqVuxgb4KY
- Response to Stimulus: https://youtu.be/Y3JlhLka4Ks