SAN DIEGO STATE
UNIVERSITY

College of Engineering, Department of Electrical and Computer Engineering

EE/COMPE 492 Senior Design

System Integration Test Report

Team HANDS-EMG

ECE Team 9 – Hand Activity Neural Detection using sEMG

EE Members:

Kelly Hubbard, Jayden Sumbillo, Kirk Young

CE Members:

Noah Marosok, Blake Pearson

Sponsor:

SDSU Department of Electrical and Computer Engineering

SDSU Smart Biomedical Systems Lab

Kelly Hubbard
Electrical Engineering

Jayden Sumbillo
Electrical Engineering

Kirk Young
Electrical Engineering

Noah Marosok
Computer Engineering

Blake Pearson
Computer Engineering

Dr. Hakan Töreyin
PI & Advisor

Prof. Barry Dorr
Advisor

Executive Summary ... 4
Updated System Description ... 5
Triage Tests ... 8
    Microcontroller SWD Flashing ... 8
    Power Supply Rail ... 9
    Machine Learning Model ... 9
    User Interface ... 9
Test Plan ... 10
Description of Tests and Results ... 12
    User Interface ... 12
    Machine Learning Classification ... 15
    Power System Wiring ... 17
    AFE Signal Output ... 19
Root Causes of Failures ... 21
    Failure in AFE Signal Acquisition ... 21
    Failure in Power System Wiring ... 21
    Failure in Machine Learning Model ... 21
Design, Specifications, or Project Changes ... 23
Updated Schedule ... 24

Table of Contents

FIGURE 1: MACHINE LEARNING PROCESS ... 5
FIGURE 2: PHYSICAL SKETCH OF DEVICE ... 6
FIGURE 3: USE CASE DIAGRAM ... 7
FIGURE 4: SYSTEM BLOCK DIAGRAM ... 7
FIGURE 5: MICROCONTROLLER SWD FLASH TRIAGE SETUP ... 8
FIGURE 6: SYSTEM TEST PLAN ... 10
FIGURE 7: FINAL SYSTEM TEST PROCEDURE ... 11
FIGURE 8: UI BLOCK DIAGRAM FOCUSED ... 12
FIGURE 9: SEMG GRAPHS ON UI ... 13
FIGURE 10: ML CLASSIFICATION OUTPUT ON UI ... 14
FIGURE 11: TRAINING UI DESIGN ... 14
FIGURE 12: MACHINE LEARNING BLOCK DIAGRAM FOCUSED ... 15

FIGURE 13: TEAM MEMBERS SEMG DATA....................16
FIGURE 14: SEMG DUAL PLOT BEST PERFORMING....................16
FIGURE 15: POWER SYSTEM WIRING BLOCK DIAGRAM FOCUSED....................17
FIGURE 16: PMIC OUTPUT WHEN LOADED....................18
FIGURE 17: PMIC 2.5V LINES WHEN LOADED....................18
FIGURE 18: PMIC 3.3V LINE WHEN UNLOADED....................19
FIGURE 19: AFE BLOCK DIAGRAM FOCUSED....................19
FIGURE 20: AFE UNEXPECTED CURRENT DRAW....................20
FIGURE 21: UPDATED TEAM SCHEDULE....................24

Table of Figures

TABLE 1: DESIGN, SPECIFICATION, OR PROJECT CHANGES TABLE....................23

Table of Tables

Executive Summary

The HANDS-EMG device is a wearable, battery-powered surface electromyography (sEMG) system designed to classify hand movements in real time using embedded machine learning. This report outlines the results of the system integration testing, highlights critical outcomes, and documents key changes made to hardware, firmware, and system design in response to test findings.

Integration testing was conducted to verify subsystem interoperability and evaluate overall system performance under realistic operating conditions. Testing covered signal acquisition via the ADS1299 analog front-end (AFE), embedded classification via the STM32 microcontroller, and power delivery through the onboard power management integrated circuit (PMIC).

The AFE signal acquisition test failed. Investigation revealed oversight in the PCB layout routing that led to excessive current on sensitive input pins, ultimately damaging the ADS1299 chip. This resulted in a revision to the final PCB layout, specifically within the AFE section.

The power system tests passed, but analysis revealed the PMIC was operating near our microcontroller's absolute maximum ratings. As a result, a more suitable PMIC, the LTC3586, with similar pinout and functionality was substituted to ensure stable operation with greater margin. This change required minimal redesign but improved long-term reliability.

Initial firmware used a machine learning model trained on the public NinaPro dataset. However, integration testing showed significant accuracy degradation across different users. In response, the team transitioned to a personalized model trained using STMicroelectronics' NanoEdge AI Studio. This pivot required major firmware updates to support a new activation-based classification scheme. Furthermore, it necessitated user-specific data collection to train the embedded inference engine.

All changes including PCB re-layout, IC replacement, firmware restructuring, and the shift to onboard ML training—are reflected in the updated system description included in this report.

Updated System Description

The HANDS-EMG device is a battery-powered surface electromyography sensor system designed to classify hand movements using machine learning, by interpreting the muscle activity signals captured from the user's forearm. By utilizing sEMG technology, it provides a non-invasive method to monitor and analyze the muscle signals in real time. This functionality is useful for applications in prosthetics and rehabilitation where accurate and efficient movement recognition is essential. The device is portable, and battery powered, making it a practical and reliable tool for improving accessibility.

The machine learning model is trained and implemented through the process seen below in Figure 1. Training begins with raw EMG data from our device, which is processed by the Matlab training controller to attach the classification to the data. The data and its associated classification are input into STMicroelectronics NanoEdge AI to build and evaluate the model. Once trained, the model is exported to a knowledge header file for deployment on the STM32 microcontroller. The microcontroller receives raw data from the analog front-end, processes it to determine if a sEMG activation event occurred, and uses the pretrained inference engine to classify movements and output the results in real time.

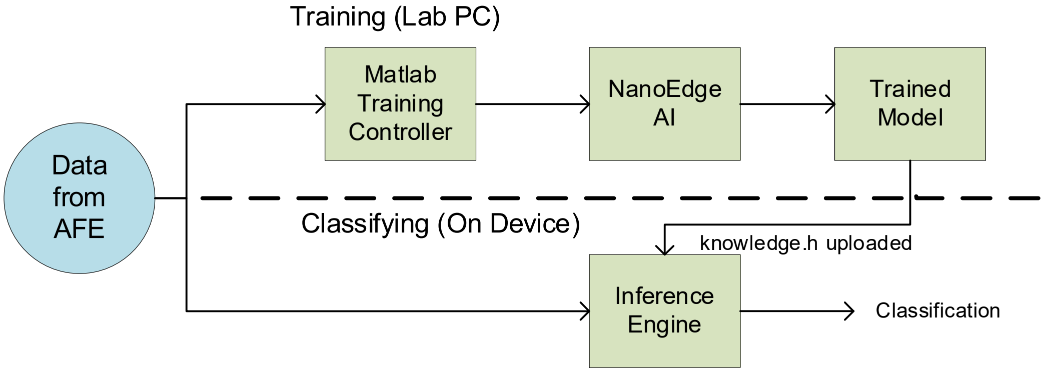


Figure 1: Machine Learning Process

Figure 1 demonstrates the machine learning process that is conducted to attain a working model on our microcontroller. The process starts at the left and works right. Notice that the training portion only occurs if the Matlab training controller is activated.

The device utilizes four channels of wet electrodes placed on the user's forearm to capture the sEMG signals. The device is designed to be arm mounted and will not exceed the dimensions of 3.25 x 3.75 x 0.75in and will not exceed a mass of 40g. A physical sketch including dimensions can be seen below in Figure 2.

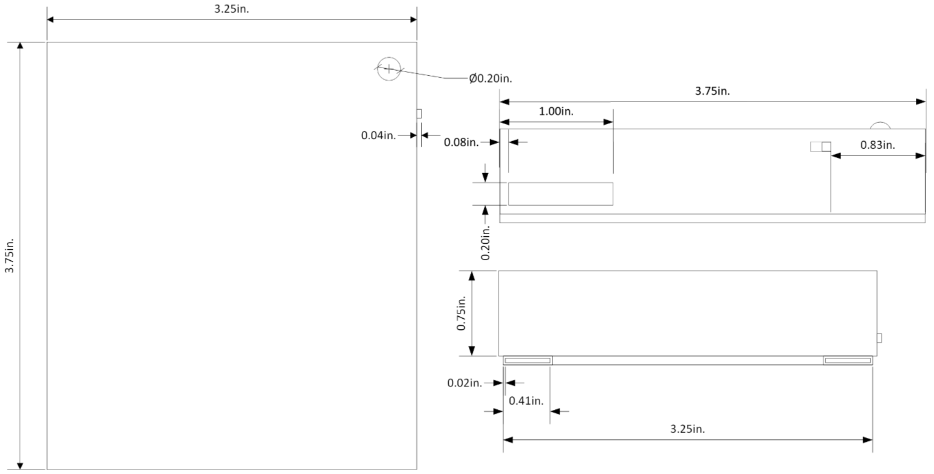


Figure 2: Physical Sketch of Device

Figure 2 shows the physical sketch of our device. This sketch is our device's enclosure that it is contained within.

Our system is comprised of three main subsystems. These subsystems are the ADS1299 analog front-end module (AFE), STM32 microcontroller, and the power management integrated circuit (PMIC). The AFE is a critical component, procured specifically for its ability to handle the complexity of sEMG signal acquisition. It amplifies and digitizes the weak bioelectric signals (typically in the range of 10uV to 10mV). The STM32 microcontroller serves as the central processor, performing feature extraction and classification using the machine learning model. The PMIC ensures stable power delivery to all components, enabling the device to operate with a 3.7V lithium-ion battery for ~38 hours. These details and the interactions between the subsystems are illustrated below in the block diagram in Figure 4.

The analog signals captured by the four channels of wet electrodes are processed through our procured analog front-end module which operates at a sample rate of 2KHz. Within the microcontroller, a pre-trained machine learning model classifies these hand movements with an accuracy of no less than 90%.

To begin using the device, the user will prepare their forearm, ensuring the area is clean and dry. Eight electrodes are placed radially around the proximal forearm, spaced approximately 2 cm apart, and positioned 2-3 cm distal from the elbow crease. A reference electrode is placed on the elbow. Once the electrodes are in place, the user runs the simulator software on a PC, connects the device to the PC via USB Micro B, and powers on the device. As the user moves their hand, the device processes the sEMG signals and transmits the classified movements to the PC, where a visual simulation displays the corresponding hand movements in real time. These steps can be seen below in Figure 3.

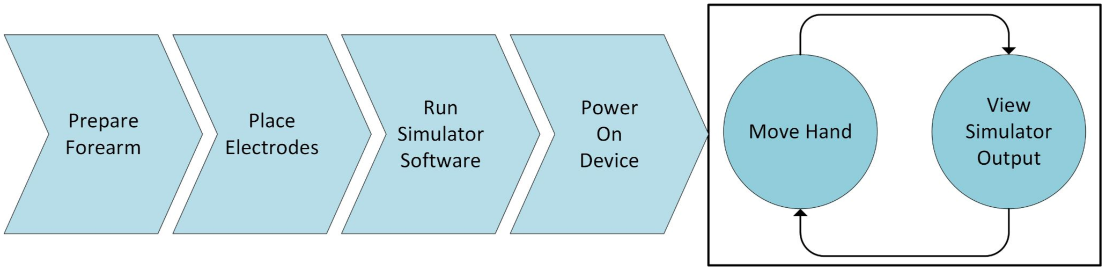


Figure 3: Use Case Diagram

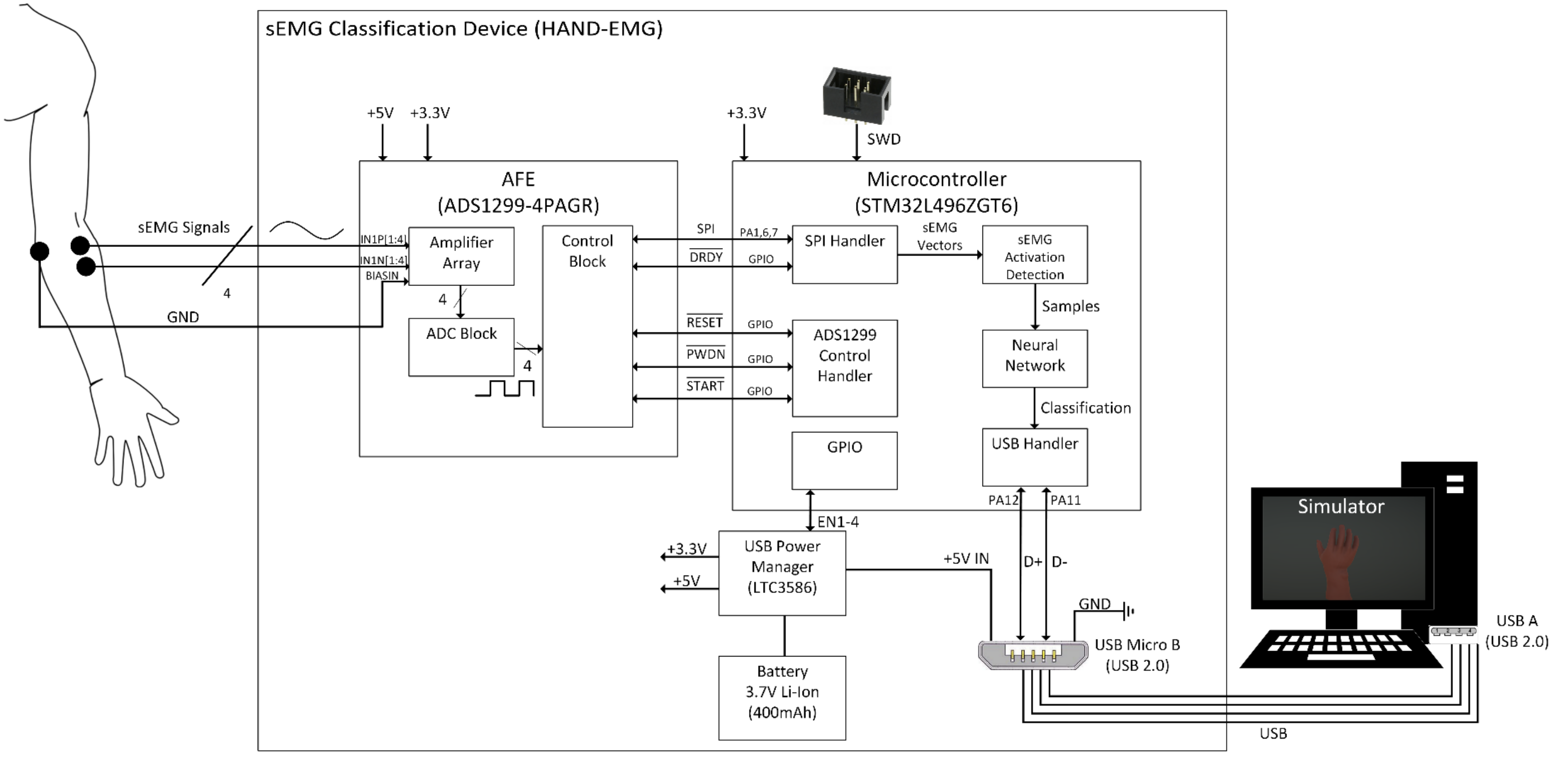


Figure 4: System Block Diagram

Triage Tests

Before formal integration testing began, a series of triage tests were conducted to quickly evaluate whether each subsystem was in a usable state. These preliminary tests focused on identifying any major failures, configuration errors, or firmware issues that would prevent meaningful test execution.

Microcontroller SWD Flashing

As shown in Figure 5, the microcontroller is connected to the *ST-Link Utility* external debugger through the following pins: PA13 (SWDIO), PA14 (SWCLK), PB3 (SWO), and NRST (NRST). Prior to testing, the first task was to wipe the STM32's on-chip flash memory to ensure any previous firmware is removed. This was accomplished simply by utilizing the debugger's built-in "Erase" function. The device's memory was verified to be successfully cleared by assessing the console output through the STM32CubeIDE program.

To test the MCU's basic functionality using the external debugger, a simple "LED blink" code was uploaded to the board, with the purpose of flashing the on-board status indicator LED at various rates. Once verified, the next test involved the actual SWD debug interface. The interface was tested and verified by stepping through the firmware.

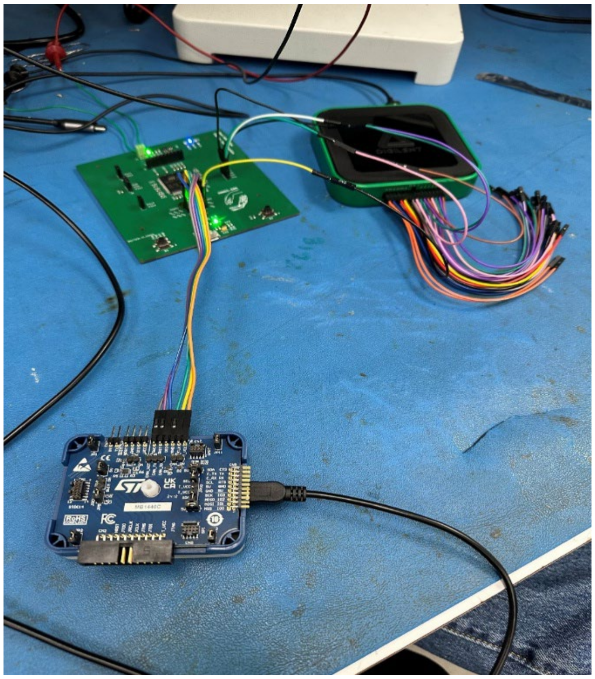


Figure 5: Microcontroller SWD Flash Triage Setup

Figure 5 shows the test setup for the simple triage test of flashing the microcontroller with the firmware via SWD.

Power Supply Rail

The power system triage tests were performed independently of any other subsystems to ensure that the power rails delivered appropriate voltages before connecting sensitive components. This safeguarded against component damage and unnecessary complications during full integration testing.

To perform this triage, the team applied power to the power management board using a 3.7V LiPo battery. With the battery applied, the team measured and confirmed that each voltage rail was reading at the correct level (3.3V, 2.5V, -2.5V). Power subsystem passed triage. All voltage rails were stable and within specification when unloaded. Battery operation and charging were also confirmed to function correctly.

Machine Learning Model

Triage testing for the machine learning classification system was performed to ensure that the microcontroller could successfully receive and run a machine learning model, and that the firmware was able to load the inference engine without crashing. This was critical before testing our model, as we knew the microcontroller's capabilities.

To perform this triage, the team uploaded a pre-validated test/demo model to the STM32 using the SWD interface. The device was run and checked for a successful boot, ensuring no resets or hangs occurred. The model was called for initial verification, and the system was monitored for stability and correct demo output. The ML subsystem passed the triage. The test model was successfully uploaded and initialized. The device remained operational, confirming readiness for live classification with our custom model.

User Interface

The purpose of this triage test was to confirm that the simulator UI could receive and plotting serial data, and that the classification display system functioned as intended. This helped ensure that later tests using live EMG input and ML outputs would be accurately visualized.

To perform this triage, a simple python script to send simulated data over the COM port in the expected format was created. When running the simulator UI and the python test script the team observed the graphing output of the UI and confirmed the waveforms were plotted correctly. Thus, the UI passed its triage testing as the graphs updated in real time and classification labels were displayed as expected when simulated data was sent.

Test Plan

The diagram below in Figure 6 shows the testing plan that is being followed. The plan begins with simple functional tests and then leads to more complex tests. The test plan concludes with the final system test and the procedures for the final system test can be seen in Figure 7 below.

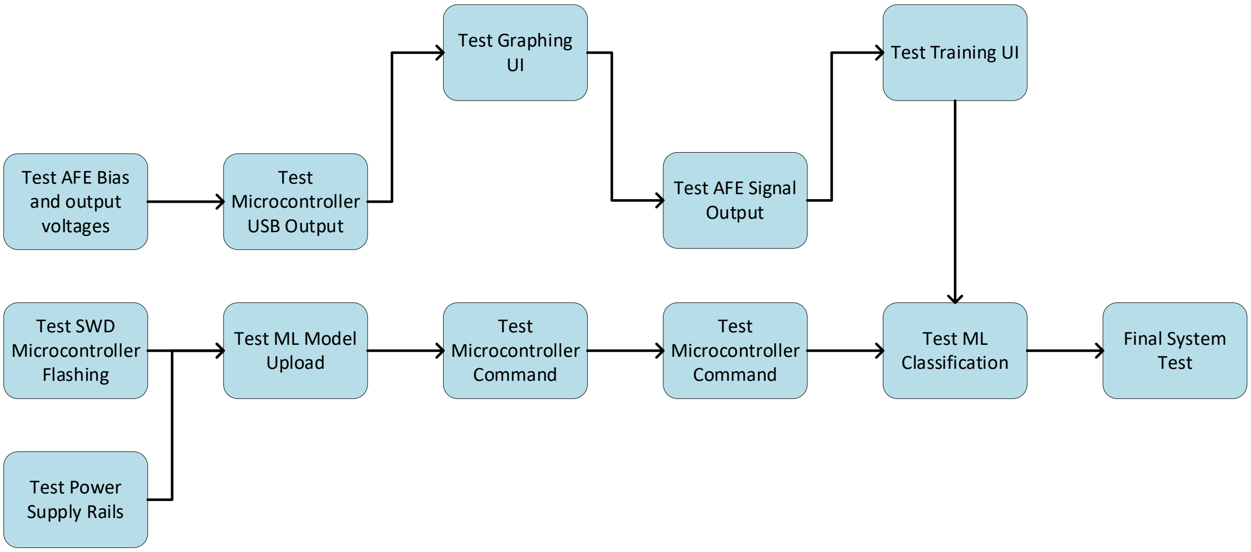

Test Graphing UI
Test Training UI
Test AFE Bias and output voltages
Test Microcontroller USB Output
Test AFE Signal Output
Test SWD Microcontroller Flashing
Test ML Model Upload
Test Microcontroller Command
Test Microcontroller Command
Test ML Classification
Final System Test
Test Power Supply Rails

Figure 6: System Test Plan

Figure 6 shows the testing plan that is followed for our integration of our device. It begins with simple functional tests and leads to more complex tests.

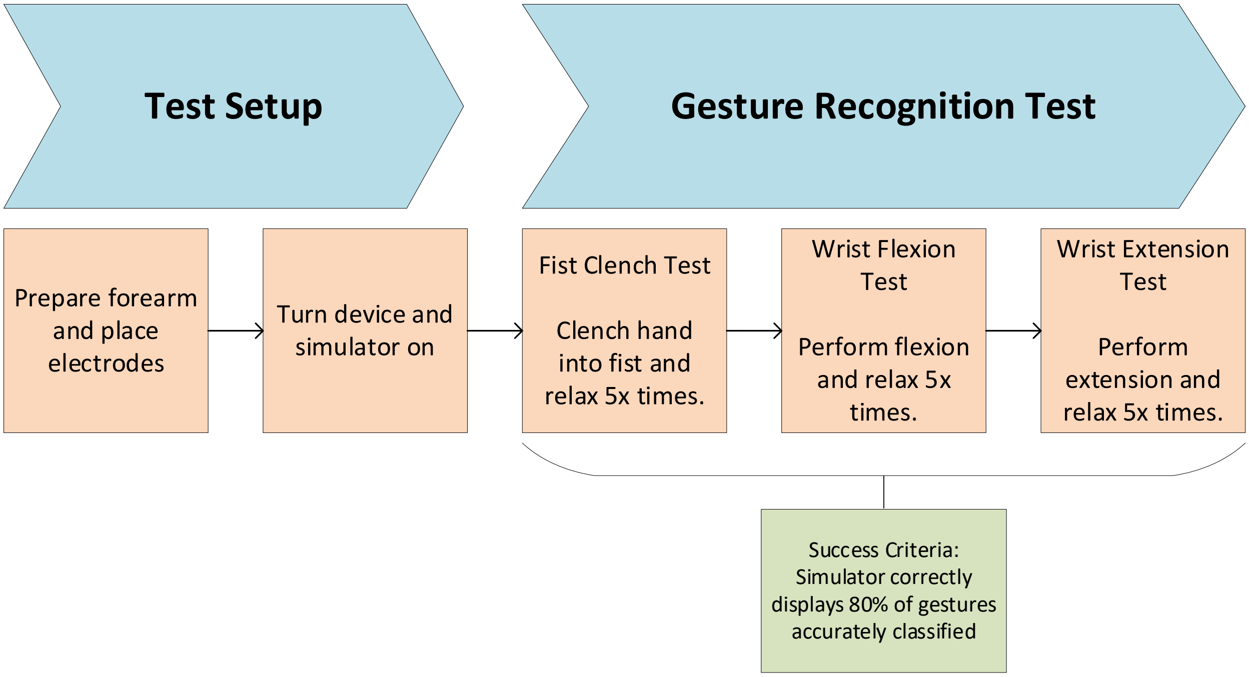

Test Setup
Gesture Recognition Test
Prepare forearm and place electrodes
Turn device and simulator on
Fist Clench Test
Clench hand into fist and relax 5x times.
Wrist Flexion Test
Perform flexion and relax 5x times.
Wrist Extension Test
Perform extension and relax 5x times.
Success Criteria: Simulator correctly displays 80% of gestures accurately classified

Figure 7: Final System Test Procedure

Figure 7 shows the testing procedure to conduct the final system test that serves as the definition of complete for our device.

Description of Tests and Results

The following is a description of all the major tests that were performed during system integration.

User Interface

The simulator and user interface (UI) portion of the system was tested to verify its ability to correctly display sEMG data received over a serial COM connection (via Micro-B USB) and properly associate user movements with the output of the embedded machine learning model. This test also proves that the training UI can properly set up the data to be processed by the ML trainer. This subsystem corresponds to the user-facing/simulator portion of the block diagram, see Figure 8.

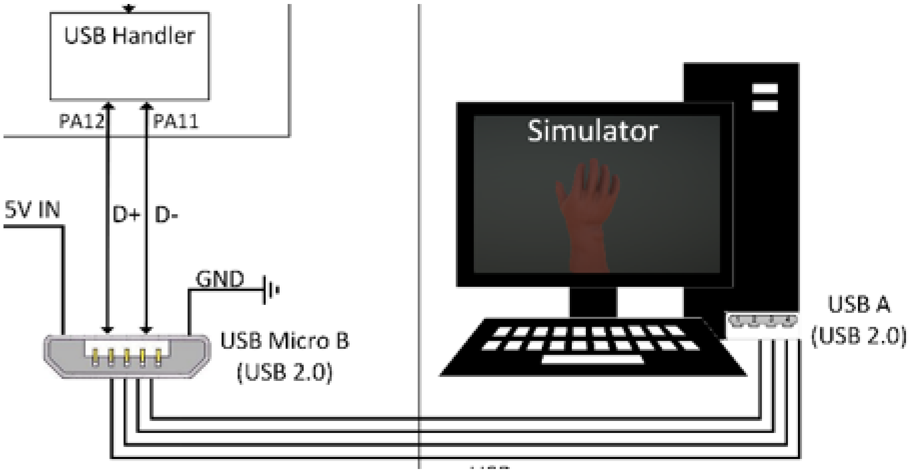


Figure 8: UI Block Diagram Focused

Figure 8 shows the block diagram zoomed into the UI portion of the device.

User Interface Test Procedure

1. Connect the HANDS-EMG device to a PC using a USB Micro B cable.
2. Launch the simulator and verify that the main user interface appears.
3. Enter Normal Mode and check that real-time EMG plots are generated from the COM port data stream.
4. Move the hand through different positions and verify that classification labels are displayed and updated at least once per second.

5. Switch to Training Mode and follow the movement prompts. Ensure that the training UI collects and saves incoming data with the correct movement label.
6. Review the output files and confirm the correct format and label tagging.

User Interface Test Results

All UI tests passed. The EMG signals were plotted in real-time with a smooth update rate. Movement classification labels updated reliably and reflected microcontroller output. The training interface is properly labeled data buffers per movement. Raw training files were verified manually. Since these are large CSV files, they are not visualized in this report. These results can be seen below in Figure 9, Figure 10, and Figure 11.

User Interface Test Figures

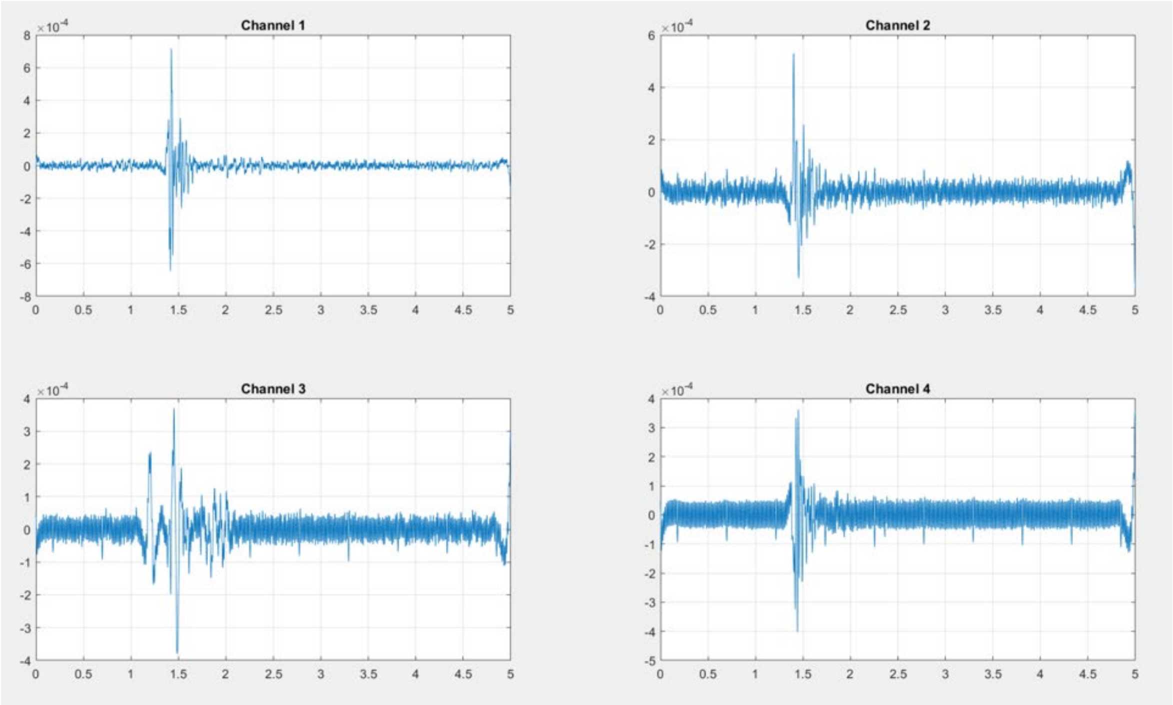


Figure 9: sEMG Graphs on UI

Figure 9 demonstrates that test 3 was successfully completed. It shows the graphs of the EMG waveforms collected by the AFE eval kit, transmitted to the custom uC board, and then transmitted to the PC via USB.

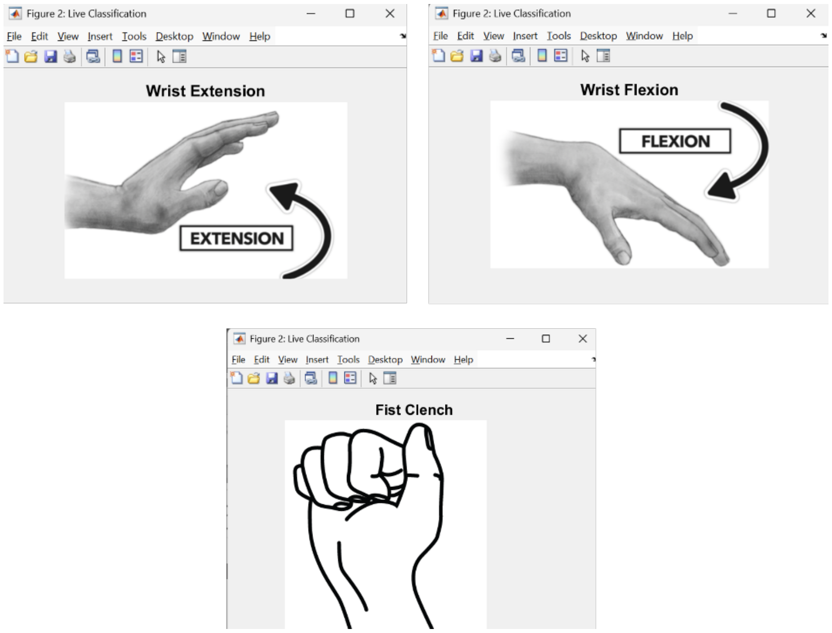
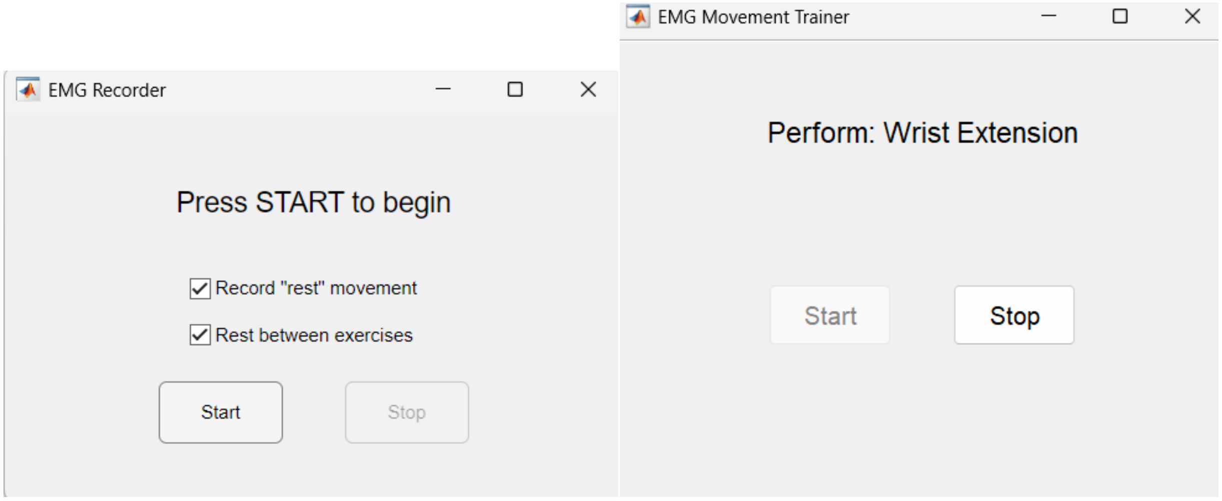


Figure 2: Live Classification
File Edit View Insert Tools Desktop Window Help
Wrist Extension
EXTENSION
Figure 2: Live Classification
File Edit View Insert Tools Desktop Window Help
Wrist Flexion
FLEXION
Figure 2: Live Classification
File Edit View Insert Tools Desktop Window Help
Fist Clench

Figure 10: ML Classification output on UI

Figure 10 shows the possible outputs for the ML live classification UI. Its important to note that these were gathered from real ML classifications from the new working machine learning model after retraining.



Press START to begin

☑ Record "rest" movement

☑ Rest between exercises

Perform: Wrist Extension

Figure 11: Training UI Design

Figure 11 shows the training UI design. When the start button is pressed on the training UI, it begins to collect data from the microcontroller. The user will perform the movement that is displayed and the data that is acquired during the movement is recorded to a CSV file related to the movement (e.g. wrist_extension.csv).

Machine Learning Classification

This test evaluates the performance of the Ninapro trained ML model with custom validation data as received by the AFE via the SPI connection. This is to verify that the Microcontroller will be able to accurately predict the gesture of a given waveform when deployed. The subsystem corresponds to the machine learning and feature extraction portion which can be seen below circled in red in Figure 12.

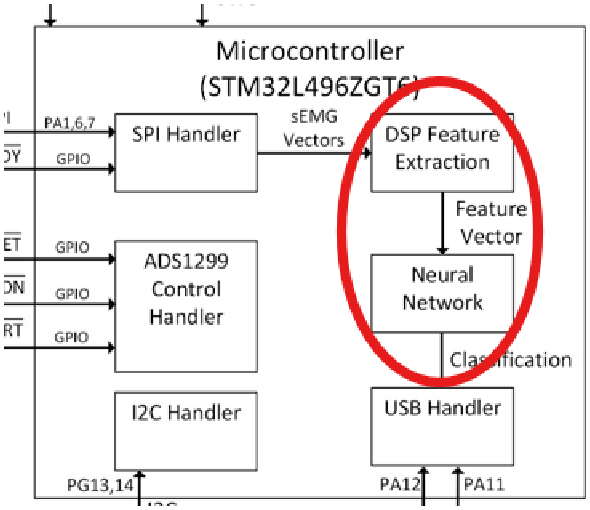


Figure 12: Machine Learning Block Diagram Focused

Figure 12 shows the block diagram zoomed into the ML portion of the device.

ML Classification Test Procedure

1. Analyze the final model for classification accuracy of data transmitted from AFE.
2. Record 3 repetitions of the five gestures selected using the AFE and transmit the waveforms to the STM32.
3. Package and clean the data in accordance with the Ninapro training data.
4. Run an inference test on all windows of data to verify that the accuracy is within an acceptable margin from training data.

ML Classification Test Results

After successfully porting the AFE, Figure 13 shows the test as it was performed. When this data was packaged utilizing the custom window function that has been used throughout this project and run through a testing of the ML model on the PC, the model produced a 54.01% accuracy. For comparison, subject 9 which had a 94.08% accuracy is plotted next to Noah’s sEMG data below in Figure 14.

ML Classification Test Figures

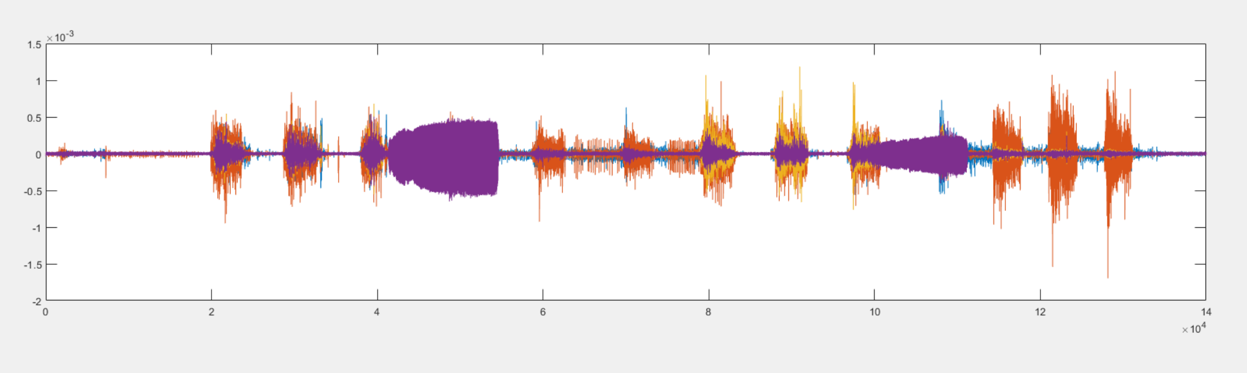


Figure 13: Team Members sEMG Data

Figure 13 shows the entire recording of Noah's sEMG data prior to windowing and feature extraction.

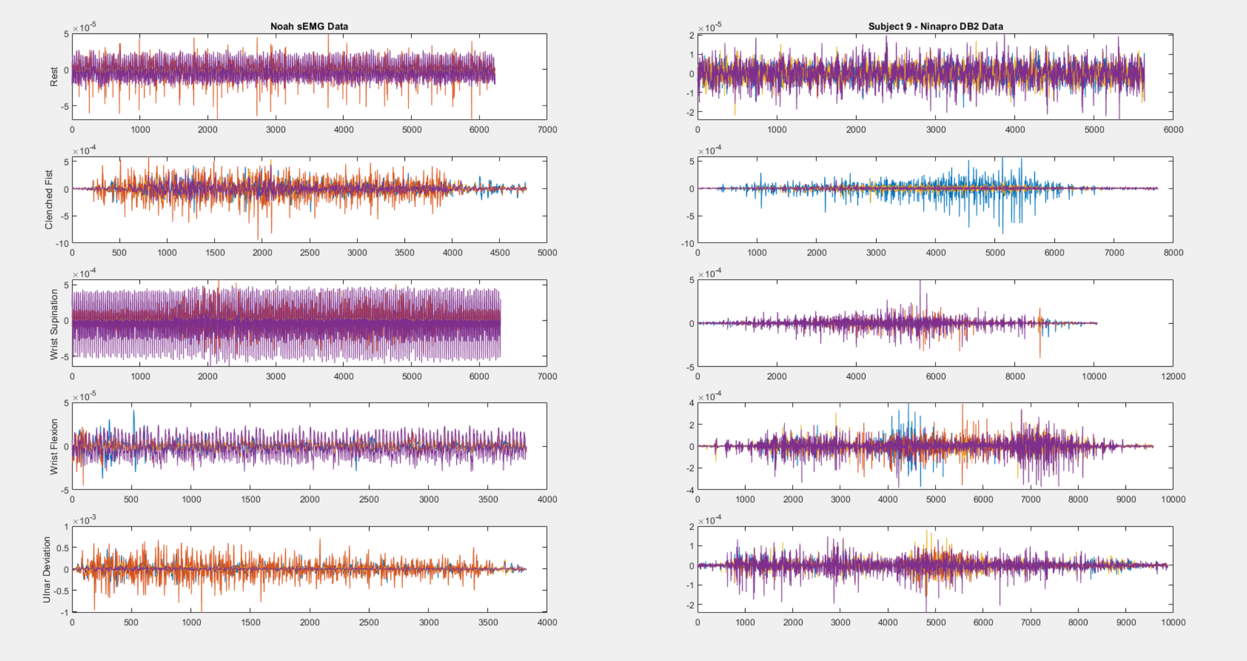


Figure 14: sEMG Dual Plot Best Performing

Figure 14 shows a dual plot of each gesture for the best performing Ninapro subject and Noah's sEMG data. Subject 9 from the Ninapro database provided a 94.08% accuracy utilizing leave-one-out-cross-validation while Noah produced a 54.1% accuracy after testing. Note the visible periodic interference in the purple waveform of Noah's sEMG data that was identified as 60Hz noise.

Power System Wiring

This test ensures that the power management system reliably provides the required voltage rails to the microcontroller and analog front end. This subsystem is represented in the power system portion of the block diagram, seen below in Figure 15.

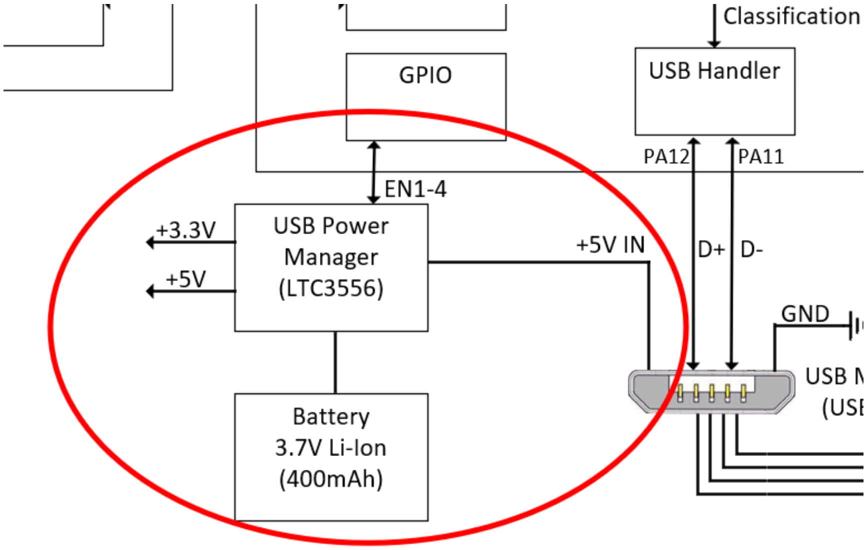


Figure 15: Power System Wiring Block Diagram Focused

Figure 15 shows the block diagram zoomed in on the section that is being tested in this test.

Power System Wiring Test Procedure

1. Power the board with no load connected and verify output rails using a multimeter.
2. Connect the AFE board and verify that it receives the proper voltage and the PMIC remains operational.
3. Connect the STM32 board and verify that it powers on successfully.
4. Measure all major voltage rails (3.3V, +2.5V, -2.5V) with all devices connected.

Power System Wiring Test Results

Tests 1 and 3 passed. The STM32 microcontroller operated correctly when powered by the PMIC. Test 2 failed, the AFE failed to power on due to a PCB layout issue. This issue led to a change in the AFE routing and is discussed further in the failure section. Test 4 also failed, the PMIC 3.3V rail spiked to 3.6V when connected to the STM32. While technically this is allowable due to the absolute maximum rating of the STM32 being 3.6V, the team decided to change the PMIC to avoid future issues. This change is reflected in the system description. These results can be seen below in Figure 16, Figure 17, and Figure 18

Power System Wiring Test Figures

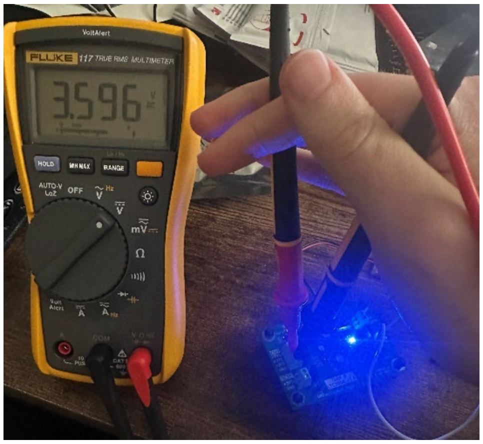


Figure 16: PMIC Output when Loaded

Figure 16 shows the PMIC outputting 3.6V when it is under load from the microcontroller. Demonstrating the 3.6V spike that caused concern on the team leading to a system description change.

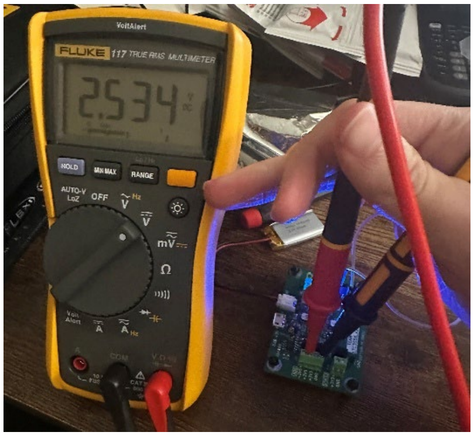


Figure 17: PMIC 2.5V lines when Loaded

Figure 17 shows the 2.5V rails working correctly when loaded.

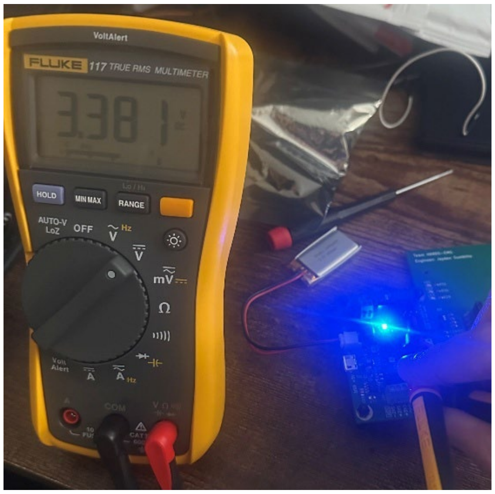


Figure 18: PMIC 3.3V Line when Unloaded

Figure 18 shows 3.3V rail working correctly when the device has no load.

AFE Signal Output

This test evaluates the operation of the ADS1299 analog front end by verifying current draw, reference voltages, SPI communication, and sEMG signal output. This functionality is shown in the AFE section of the block diagram, seen below in Figure 19.

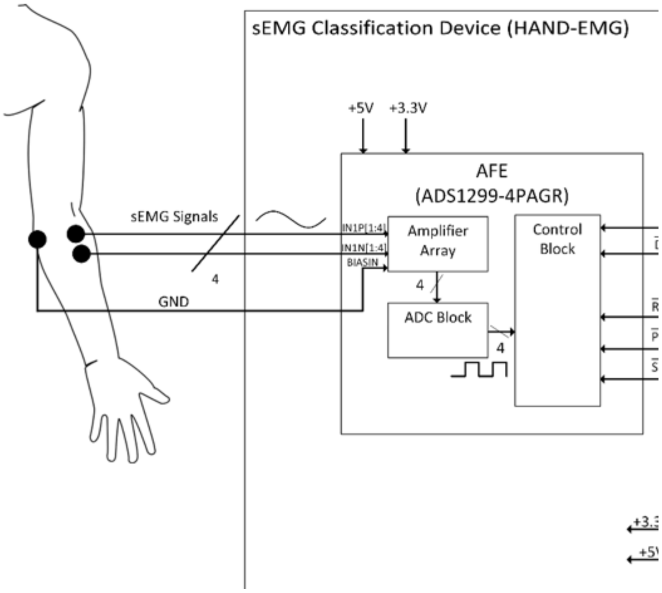


Figure 19: AFE Block Diagram Focused

Figure 19 shows the block diagram zoomed in on the subsystem that is being tested in this section.

AFE Signal Output Test Procedure

1. Power the AFE using external power supplies and measure current on the analog and digital lines.
2. Probe the AFE's reference and bias test points using a multimeter.
3. Connect the AFE to the STM32 via SPI and attempt to initialize communication.
4. If successful, stream AFE data to the simulator and observe real-time plots.

AFE Signal Output Test Results

All tests failed. The digital power consumption was ~107mA, far above the 0.1mA absolute maximum. Bias voltage was 0.9V (expected 4.5V). Reference voltage was correct at -2.5V. Communication failed and the AFE was unresponsive and did not transmit data. These results can be seen below in Figure 20.

AFE Signal Output Test Figures

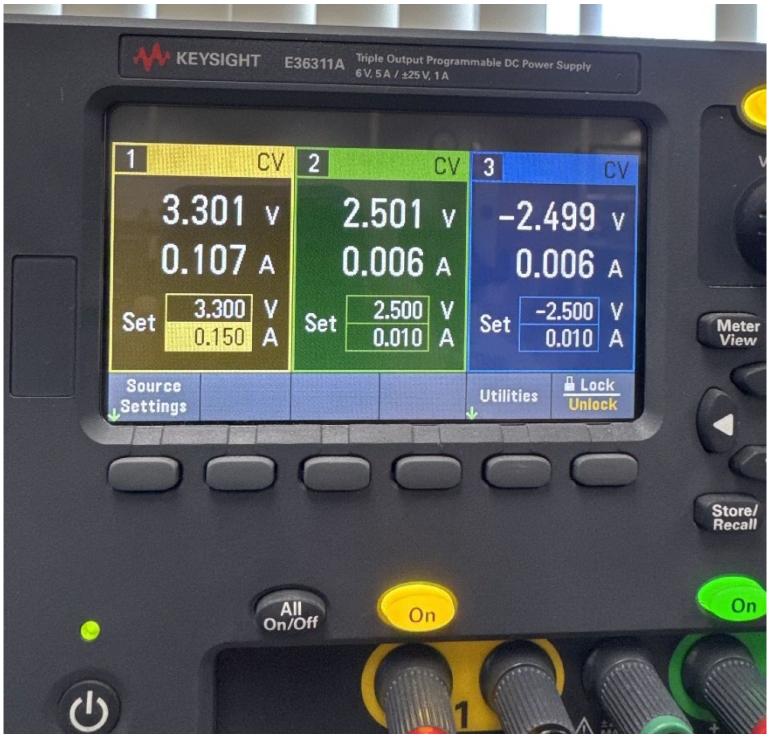


Figure 20: AFE Unexpected Current Draw

Figure 20 shows the unexpected current draw from the AFE device. 107mA is 106.9mA over the expected current limit. This figure demonstrates that test 1 fails and causes all subsequent tests to fail.

Root Causes of Failures

Failure in AFE Signal Acquisition

When designing the ADS1299 board, an EEPROM section was proactively included in the design with the idea that we would have it there if we found we needed it while testing the board's functionality. This EEPROM had resistors that connected the digital power to ground, however 0 ohm resistors were used instead of those with small values for testing purposes, leading to a short. This short caused a large current draw from the digital power supply to the ADS1299 chip, which we found damaged it. It was determined the chip was damaged after observing that the correct voltage would only be received after a large amount of current to initially power the chip was sent. After trying to connect to the microcontroller to see if anything would be sent, only bits of information could be sent and not at the correct time. This failure was mitigated with a simpler and more concise design. We removed a portion that we found were not going to be needed, including the external clock, EEPROM chip, ensuring we only had the filtering portions for the input power and signals. This was also backed up by similar designs that followed the same specifications with working models for the final design.

Failure in Power System Wiring

When we designed the PMIC, we designed it with the intention of getting three separate outputs; +2.5V, - 2.5V, and 3.3V. The 3.3V line was measured to be 3.7V when connected to our microcontroller. While this is acceptable, it put the voltage near the absolute maximum rating, therefore we decided to change the PMIC. This was due to the complexity of the buck converter we were using, as well as the chip. In order to fix this, we decided to switch from the LTC3556 to the LTC3586, which is a nearly identical chip except that it comes with a boost to +5V. This way we could have a 3.3V output for the digital inputs and have a 5V rail that supplies a separate power portion for the AFE portion. This portion now has a conversion for a –5V that gives a fixed low noise –2.5V from an LDO, and the +5V to another LDO giving a 2.5V. This now provides an accurate and consistent differential 2.5V supply to the AFE portion and a stable 3.3V to the digital supplies.

Failure in Machine Learning Model

When the machine learning task was laid out initially, it seemed feasible that some form deployed model would be able to accurately predict a subset of the gestures. As stated in the results, however, the actual performance was far below what would be deemed acceptable. For this, several reasons can be attributed. For one, most classification tasks utilizing the Ninapro dataset all 8 trans radial sEMG channels, as well as four additional channels to classify hand gestures are being used while the proposed design is only meant for 4 channels. Due to this, the official Ninapro acquisition procedure includes no mention specific trans radial electrode placement. Thus, the matching between the database channels and the electrode placement is truly unknown. Additionally, Ninapro database was captured using very advanced bioelectric capture equipment that is thoroughly robust to noise, while our data acquisition contained much less noise filtering capabilities. Thus, a clear 60Hz interference is visible in Noah’s sEMG data. Finally, with biopotential signal measurement, differences in even demographically similar subjects can be vastly different. Differences such as age, muscle mass, body fat, daily water consumption, dead skin on the surface of the skin, or oil from sebaceous glands can interfere with proper signal acquisition. Due to these reasons, a personalized model from each subject that wishes to use the device must be made. This is done by performing a basic training loop, then the simulation can begin, and the device is ready to use.

Design, Specifications, or Project Changes

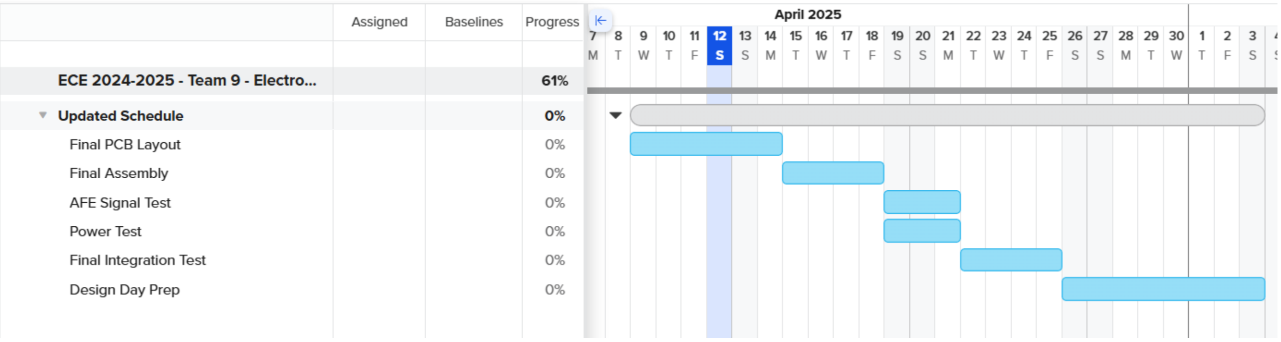
After extensive analysis of the system's failures, the team made multiple changes to the system description and design. The changes are listed in Table 1 below.

Table 1: Design, Specification, or Project Changes Table

Subsystem	Change	Reason
AFE	More concise design around the AFE. EEPROM chip and multiple Op Amps removed. Multiple configuration resistors removed.	The team's previous failure was a PCB routing issue that became apparent after connecting to power. A more concise design prevents this issue by removing the parts related to its cause.
Machine Learning	Transitioned to a personalized model trained using NanoEdge AI Studio. Update firmware to support activation-based classification scheme. Data collection from users to train the embedded inference engine.	The model based on the NinaPro data set was found to be inaccurate during testing as shown above. A personalized model trained on our data allows us to control the data collection and movements used.
Power	PMIC was changed from LTC3556 to LTC3586. A separate LDO was added to the -2.5V line.	The LTC3586 enables better control over the voltage regulation to prevent the issue with the voltage spike while loads are connected.

Updated Schedule

Due to the failures during our system test and the resulting changes to our design our schedule was updated to accommodate a new PCB order and testing of that resulting PCB. The Team Gantt chart below in Figure 21 shows the updated schedule.



	Assigned	Baselines	Progress
ECE 2024-2025 - Team 9 - Electro...			61%
▼ Updated Schedule			0%
Final PCB Layout			0%
Final Assembly			0%
AFE Signal Test			0%
Power Test			0%
Final Integration Test			0%
Design Day Prep			0%

Figure 21: Updated Team Schedule

Figure 21 shows the team's updated schedule as the semester ends. Within the next 15 days the final PCB will be ordered, tested, and prepared for senior design day.