Boston University
Electrical & Computer Engineering
EC464 Senior Design Project

Second Prototype Testing Report

Integrated Laser and Electronic Model for Photoacoustic Neural Stimulation

By Team 19
Team Members

Tian Huang tianh1@bu.edu
Raphael Mok rmok@bu.edu
Ian Lee ianxlee@bu.edu
Mateus Toppin mtoppin@bu.edu
Viet Nguyen vietng99@bu.edu

Equipment and Setup:

The equipment and setup for our second prototype test were aligned with the requirements detailed in our test plan. Utilizing an Arduino Uno as the core component, our setup also included an oscilloscope, BNC to alligator clip wires, M/M jumper wires for secure connections, USB port, and a computer to run the Python-based software application designed for parameter adjustments. This setup aimed to facilitate a seamless interface between the hardware and software components, ensuring the generation of precise pulse waves for neural stimulation purposes. Overall, the Arduino device is interconnected with a computer device through a USB and the Arduino is hooked up to the Oscillatorscope for the purpose of testing with alligator clips. The jumper wires are connected to pin 9 of Arduino and the ground. The process would be simply uploading input parameters from the software to the Arduino for pulse generation. The Arduino will take it and use a serial method to output the desired wave.

The primary objective was to verify the system's capability to generate pulses with specified parameters input through the software application, focusing on frequency, duty cycle, and pulse width modulation accuracy. The setup aimed for seamless interaction between the software application and the hardware, ensuring accurate pulse generation on the oscilloscope. As described above, the hardware side was set up exactly as described in the test plan. The same could be said for the software side where GitHub stores all our code and the team initializes all components before the test starts. Overall, it is done as said in the test plan and consistent with it.

Discuss Detailed Measurements taken:

Throughout the testing phase, our team conducted a series of measurements focusing on the accurate generation of pulse waves based on specified parameters. These parameters included variations in frequency, duty cycle, and pulse width modulation, which were inputted through the software application.

Initial Setup and Baseline Measurements: The initial test began with setting the frequency at 1kHz and duty cycle at 50%, observing the pulse generation on the oscilloscope to establish a baseline for accuracy and consistency. Parameter Variations and Sequence Testing: Subsequent tests involved altering the frequency to 0.5kHz, 100Hz, and 10Hz, while adjusting the duty cycle to 75%, 25%, and 30% respectively. The output was closely monitored for each set of parameters to ensure it matched the expected results. Particularly notable was the sequence test at 10 Hz and a 30% duty cycle, where the sequence {200,100,50,150,100} was inputted. The timing for this sequence was meticulously recorded, confirming the system's ability to accurately manage on and off sequences according to the provided specifications.

Advanced Frequency and Timing Analysis: A critical test involved setting the frequency at 1.7kHz with a sequence of {15000,8500,10000} to evaluate the system's precision in handling higher frequency ranges and complex timing requirements. The results from this test provided insightful data on the system's capability to deliver precise control over pulse generation, with times closely matching the expected durations of 8.8, 5, and 5.9 seconds.

Lastly, the software application works as intended. One of the sequences we tested was with a PWM sequence of {1000, 2000, 1000, 2000}, frequency of 1700 Hz, and duty cycle of 30%. The displayed images on the oscillator scope match what we inputted in the software since the square pulse definitely seems about 30% out of a full wave. The sequence is on for about 1 ms, off for 2 ms, on again for 1 ms, and off for about 2 ms. Based on the calculation, we should get about 10.2, or precisely, 10 pulses should be shown in the oscillator scope. Which we did. Thus, the measurement we intended is overall, met.

Conclusion:

The second prototype testing phase has proven to be valuable, offering a detailed insight into the capabilities and areas for improvement in our system. While the software application demonstrated good performance in parameter management and user interface efficiency, the testing also highlighted critical limitations in hardware performance, particularly concerning the precision of pulse sequence generation. Despite the software's accurate parameter adjustments, discrepancies in pulse outputs, such as missing or extra pulses, indicate a need for hardware reassessment.

Key Findings:

- Software Performance: The Python-based application effectively interfaced with the Arduino, allowing for precise input of frequency, duty cycle, and PWM parameters. The user interface was user-friendly and facilitated easy adjustments and monitoring.
- Hardware Limitations: The Arduino Uno, while fundamental in our setup, showed limitations in its capacity for high precision and control necessary for our application. This was evident in the irregular pulse sequences observed during the tests.

Future Directions:

- Hardware Upgrade: It is imperative to explore alternative microcontroller units (MCUs) that offer higher performance and precision in timing and control. This could involve comparing specifications, processing capabilities, and community support for different MCUs.
- Software Enhancements: Incorporating a waveform preview feature within the software application could significantly improve usability and accuracy. This addition would allow users to visualize expected outputs before actual pulse generation, facilitating better experiment planning and setup.

Overall, as discussed in the measurement section, the software application along with the pulse-generating algorithms certainly works with the current Arduino. However, the major problem is with the limitation of the hardware, specifically, the Arduino clock rate might be too slow. However, as with the test itself and measurement taken, like with a sequence of {1000, 2000, 1000, 2000}, 1700 Hz, and 30% duty cycle, the output is what we expect and the team concludes the algorithms for pulse generating and software/hardware interface are all implemented correctly. Since the resulting 10 generated pulse is what we expected. However, it is important to note that the timing works accurately for certain test cases but not for the others. Like the case with the frequency at 1700 Hz and a sequence of {15000, 8500, 10000}, we met the required result of 8.8, 5, and 5.9 seconds. But, for other random timing, especially, with long pulse chains, we start to get inconsistency with gaps between sequences and pulses as well as some cut-off of pulses. Right now, this is deemed as a hardware limitation issue.

In conclusion, the insights gained from this testing phase have laid a solid foundation for the next steps in our project's development. By addressing the hardware limitations and enhancing the software interface, we aim to achieve a system that not only meets but exceeds the precision requirements for neural stimulation applications. Continued collaboration, research, and testing will be essential in realizing these improvements and ensuring that our final product stands.