Boston University
Electrical & Computer Engineering
EC464 Senior Design Project

Second Prototype Testing Plan

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

Required Materials:

Hardware:

- Arduino Uno
- Arduino Uno USB port/connector
- Oscilloscope
- BNC to alligator clip wire
- M/M Jumper wire
    - Output pin 9 and Ground
- Computer device

Software:

- Arduino Code
    - Uses Arduino IDE
    - Timer 1 library extension
    - Serial library
- Python
    - Tkinter library
    - Sys library
    - Warning library

Objective:

- Arduino hardware will generate the needed pulses (for neural stimulations) on the oscillator scope. The needed parameters like frequency, duty cycle, pulse width modulation will be inputted from the software App. Thus, the software application will act as an input and Arduino will read in the value for the pulse.

Set Up:

The main set up of the second prototype test is relatively straightforward. The first step is to turn on the oscilloscope and adjust the parameters such that we can see the output pulse on the display screen. The next step is to connect the Arduino Uno board with a computer through the USB connector so that the software application will upload data to the Arduino Uno. The user will input data like frequency, duty cycle, and other parameters on the software application, then the software application will write these data onto the Arduino in a specific format suggested by the Arduino Uno. The two will interact and continuously update the pulse on the oscilloscope. The next major setup is connecting the male to male jumper wire at ground pin and pin 9 to the oscillator scope using banana clips. The exposed portion of the jumper wires are then clipped to the alligator clips, with red corresponding to the PWM output and black corresponding to the ground pin. From there, the BNC connector is attached to the oscilloscope and a quick preliminary test is run to auto-scale to the current scale used for the frequency. The voltage will remain in a constant range of 0 to 5 volts and so the vertical axis of the oscilloscope should not need to be adjusted after the first adjustment. Lasly, the software application needed to be set up and running for this to work. The software will be run using Python through the terminal.

Pre-testing Setup Procedure:

Hardware side:

- Connects Arduino to a PC
- Hooks Arduino to the oscillator scope through M/M jumper and alligator clip
- Scale the oscilloscope as see fit
- Change setup if needed for different parameters (frequencies, PWM, duty cycle)
- Ensure Arduino pins are correctly connected

Software side:

- Load up code from Github
- Run the main file to get the software application running
- Run and upload the Arduino IDE code to the Arduino
- Initialize parameters
- Change code if needed for more variety test
- Uses Gitbash to run the software

Testing Procedure:

1. Enters appropriate parameters in the Software so the desired behavior is shown in the oscilloscope
2. Set the frequency to 1kHz, the duty cycle to 50% as an initial test, and the sequence to one number measuring anything above 30k to allow for significant time running, and observe the output to be identical to the above parameters
3. Change this variable frequency to 0.5kHz, 100Hz, and 10Hz, while also altering the duty cycle to 75%, 25%, and 30% and ensure the output matches the specified parameters
4. Alter the sequence while still at 10Hz and a 30% duty cycle to be {200,100,50,150,100}, and set a timer while running this code to ensure the sequence follows being on for 20 seconds, off for 10, on for 5, off for 15, and again on for 10. This is to ensure that the timer works as expected.
5. Change the frequency to 1.7kHz and adjust the sequence to be {15000,8500,10000}. Once again time this and ensure the length of time for the outputs is again matching the values of approximately 8.8, 5 And 5.9 seconds. This will allow for easy insurance that the microsecond delay in the code is functional along with the millisecond one.
6. Nothing too specific, just need to make sure any entered value on the software app will display an appropriate number of pulses over the matching time.

Measurable Criteria:

The criteria for successful running and output is as follows:

I. The Arduino is able to successfully output a voltage that matches a PWM signal with duty cycle and frequency matching to what they were initially set at the start of each test
II. There is a clear time for which the output of the arduino turns off and on again assuming the sequence variable has more than 2 values in it
III. The time for which the output displays PWM is equal to the time it would take for the output to cycle through the number of pulses specified in sequence. This should also be true for the time in which the output is in the off state and no voltage is being output.
IV. The voltage output is 0 at the end of the sequence to insure no interference with other machinery when being used in actual trials.
V. The displayed pulses matches the parameters entered from the software application