Boston University
Electrical & Computer Engineering
EC463 Capstone Senior Design Project

Second Prototype Testing Plan

EEG-based Brain-Computer Interface

Submitted to:

Prof. Alan Pisano
apisano@bu.edu

by

Team 04
Brain Force

Team Members:

Alexander Johnson atjohnso@bu.edu
Brendan Shortall shortall@bu.edu
Dayanna De La Torres dayannat@bu.edu
Jonathan Mikalov jmikalov@bu.edu
Mitchell Gilmore mgilm0re@bu.edu

Submitted: 3/03/2023

Required Materials

Hardware:

- EEG Electrode Headset
- OpenBCI Cyton PCB
- USB-A to Micro-USB cable
- PIC32MX250F128B Microcontroller
- chipKIT UDB32-MX2-DIP bootloader
- ADS1296 Bio-sampling chip
- Multimeter + Probes

Software:

- OpenBCI GUI
- Arduino IDE
- OpenBCI Cyton Firmware Library
- Our Python code
    - 3D Virtual environment:
    - main.py: run simulation mechanics and control features
    - generate.py: create 3D objects and lighting for the environment

Set Up

The goal of this prototype testing is three-fold. The first and most important is to continue our active investigation into a bug in our system's firmware. We have achieved two way communication between our computer and the on-board microcontroller. We will first ensure this communication channel is working properly. The main problem with the data communication channel is that we cannot receive input from the ADS1296 bio-sampling chip. We will first test the hardware connections in and out of this chip to ensure that the problem had not been overlooked in the PCB fabrication process. From there, we will look into our firmware and systematically debug in hopes of finding the cause of this issue.

The second goal of this prototype testing is to test our software that is responsible for receiving the data stream from the device. The goal of this software is to allow us to easily begin and end the data stream with the click of a button, and automatically store this data for later use. Once we have these two systems working independently, the output of this software will be fed into our neural network in real time.

This leads to the third and final goal of this prototype testing, the neural network and virtual environment. The network is currently in development and does not have many quantifiable

parameters to test on, mainly due to the fact that we are not yet collecting our own data. Our virtual environment is also in development, but we are hoping to emulate a stream of actions that would come from the neural network in order to test the overall functionality of the virtual environment.

Set Up Procedure

Set up for testing our hardware will begin by collecting all of the materials needed to test this system. These materials are listed above. We will then begin by compiling firmware and uploading to the microcontroller that tests if we have two-way communication between the chip and our computer. If this works, we will then include firmware that retrieves data from the sampling chip. We will compile and upload this new firmware to the board, and connect to the OpenBCI GUI. If the output is as expected, we will plug into the headset and perform further tests. If not, we will debug the system to find the reason why.

Testing the software is fairly straightforward. There will be three command line functions to test, start_stream(), end_stream() and stream(n) where n is the number of seconds to stream for. For each of these functions we will make sure that data is output to the proper place, and that the data is formatted correctly.

Testing our virtual environment will need a series of test cases that emulate output from the neural network. This will likely be a command labeled 0-3 (inclusive) input into the datastream every second or so. Multiple command sequences will be tested in order to ensure that there are no bugs within the system.

Testing Procedure

The testing procedure for the PCB includes:

1. Begin by ensuring that power is being sent to the board
2. Reupload firmware
3. Test if we can create a data stream between microcontroller and computer
4. If success, try to get data from sampling chip
5. If successful, plug into the headset and test the quality of signals.
6. If successful, plug into the software script and stream data on command.
7. Else debug and repeat

The testing procedure for the software includes:

1. Loading up the python script

2. Ensure that it compiles properly and that all dependencies are installed
3. Execute the start function
4. Execute the stop function
5. Check data output
6. Execute the start(n) function
7. Check data output
8. Test on our board if possible

The testing procedure for the virtual environment includes:

1. Load virtual environment script
2. Ensure test cases fully test the script
3. Run environment with keyboard commands
4. Run a test case and make sure the environment updates properly
5. Evaluate and repeat

Measurable Criteria

- Hardware
    - Continuity between necessary pins
    - Data stream between computer and microcontroller
    - Firmware compiles?
    - Can we get data onto the computer?
- Functionality of software functions
    - Start
    - Stop
    - start(n)
- Functionality of virtual environment
    - Test cases
    - Inputs 0-3 correlating to model output

Score Sheet

Hardware:

Function:	**Works?**
Power?	
Continuity?	
Communication w/ Microcontroller?	

Communication w/ sampling chip?	
Works w/ Sensors?	
Works w/ scripts?	

Software:

Function:	**Works? (Y/N)**
begin_stream()	
end_stream()	
stream(n)	

Virtual Environment:

Test Case:	**Valid output? (Y/N)**
Test 1	
Test 2	
Test 3	