Brain-Computer Interface System Documentation

Firmware, Hardware, and Host API Guide

IntoMind Evaluation Prospectus
Round 3

August 2, 2025
Mandy Liao

I. Introduction

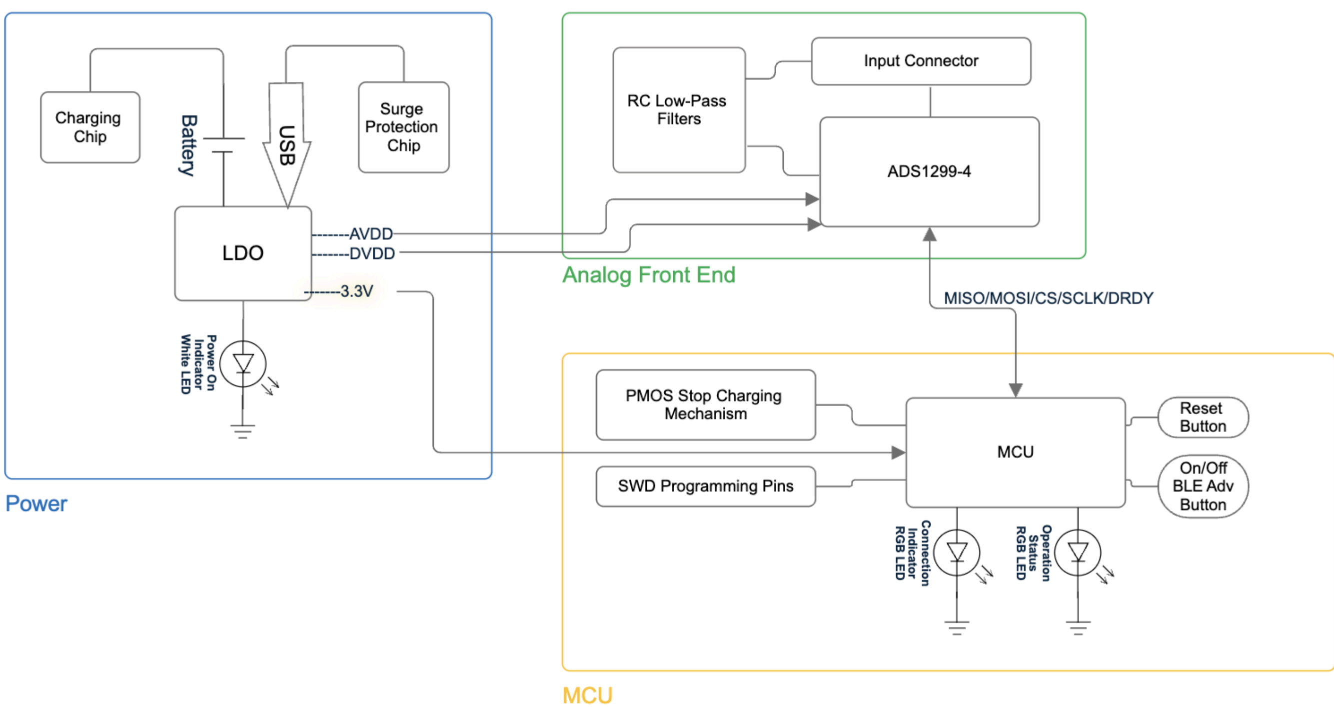
This document provides full technical documentation for operating the BCI-on-a-PCB system, including an overview of the hardware, firmware logic, and the host-side API. The goal of the system is to enable real-time, low-latency, high-reliability acquisition of raw neural data via USB or BLE from a wearable PCB platform.

Note the following safety guidelines:

- Always inspect electrodes and cables before use.
- Do not operate near water or in high-humidity environments.
- Ensure the battery is charged using only approved USB sources.
- Avoid touching the PCB during operation to minimize noise or shorts.

II. Hardware Overview

2.1 System Diagram



2.2 PCB Specifications

Dimension: 29 mm radius circular shape[1]

Connectors:

- USB-C Connector
- JST Battery Connector
- Electrode header (4 channels with common reference)

LED Indicators and Buttons (Section 2.3)

Key IC Chips:

- MCU: Raytac MDBT50Q-1MV2 (nRF52840)
- ADC: ADS1299-4

Connectivity:

- BLE 5.0
- USB (CDC ACM)

Power Source:

- Rechargeable 3.7V Lithium Polymer Battery with protection IC
- Charging via USB host and wall

2.3 User Interaction Design

Buttons	Button 1[2]	Button 2
Location	MCU GPIO	MCU GPIO
Short Press	Begin BLE Advertising	Reset MCU
Long Press (> 3s)	Power on/ off	-

[1] PCB size remains constant after modifications. All modifications are listed in the footnotes of this document

[2] Edited the PCB to have the power button controlled by MCU GPIO

LED Indicators	RGB LED 1: Operating status[3]	RGB LED 2: Connection status	RGB LED 3: Charging	White LED 4: Power
Location	GPIO	GPIO	Charging Chip	Output of LDO
Color 1	RED: Record Mode	RED: Idle State (not connected)	RED: Low Battery	WHITE: Power on
Color 2	BLUE: Livestream Mode	GREEN: USB Connected	GREEN: Battery Charging	-
Color 3	PURPLE: Passive Mode	BLUE Blinking: BLE Advertising BLUE Solid: BLE Connected	BLUE: Battery Charged	-

III. Firmware Overview

3.1 Architecture

Development Platform

- Language: C (embedded C)
- SDK: Nordic SDK 17.1.0
- SoftDevice: S140
- Target Board: PCA10056 (nRF52840-DK)
- MCU: nRF52840 (ARM Cortex-M4F, 64 MHz, 256KB RAM, 1MB Flash)

[3] Edited the PCB to add two RGB LEDs

System Architecture

Layer	Function
ADS1299 Interface	• SPI communication for EEG data acquisition
Communication	• BLE/USB interface support • Intelligent switching
Control	• API command processing • Mode management
Hardware Abstraction	• GPIO • Timers (timestamp generation) • Power management (low-power, power switching, charging)
Application	• Data processing, packaging, and transmission

3.2 Core Features

Operation Modes with Visual Feedback

Mode switching is controlled by the host API (Section 4.1). The firmware provides visual feedback, low-power management for Passive mode, and data streaming for Record and Livestream Mode.

Data Acquisition

- 4-channel EEG data capture via SPI interface with ADS1299-4
- Hardware-timestamped samples and event flag system

Dual Communication Interfaces

BLE (Nordic UART Service)

- 40ms advertising interval
- MTU negotiation for optimal throughput (up to 244 bytes)
- Connection parameter optimization for throughput and power (20-75ms intervals)
- Robust connection management (automatic reconnection and fallback advertising)

USB CDC ACM (Virtual Serial Port)

- Standard CDC class; universal driver support
- Full-speed USB 2.0; high throughput

Priority handling of USB over BLE when both are present

Control Commands

- Real-time parameter configuration with ASCII-based commands from the API

Acquisition Control	S (Start)	X (Stop)
Device Management	N<name> (Device renaming)	
Parameter Configuration	R<rate> (Sample Rate)	G<gain> (Channel gain)
Input Mode	M<mode> (Monopolar/Differential switching)	
Event Control	E<index><state> (Manage event marker)	

3.3 Technical Features

Real-time Processing

- Interrupt-Driven DRDY (Sub-millisecond response to new samples)
- Zero-Copy SPI Transfers for Direct DMA-based data acquisition
- Hardware Timestamping
- Reliable button operation through 50ms debounce handling
- Circular buffer management for efficient memory when handling data

Power Management

- Automatic power switching prioritizing USB over battery
- Automatic charging cutoff when full battery[4]
- differentiation between USB host and wall power
- Sleep mode and low-power mode

[4] Added P-MOSFET-based charging circuit to PCB

Data Packaging

Field	Size (bytes)	Description
Start Byte	1	Packet Synchronization (0xAA)
Sample ID	4	32-bit counter
Timestamp	4	32-bit Hardware timer ticks
EEG Channels	16	4 channels × 32-bit signed values
Event Flags	1	8-bit event marker bitfield
End Byte	1	Packet termination (0x55)

Error Handling

- Stack Overflow Protection
- SPI Transaction Verification
- Automatic fallback and retry mechanisms for connections
- Bounds checking for all data transfers to protect against buffer overflow
- Graceful Degradation

3.4 Firmware Specifications

Memory Usage

- Flash: ~150KB (including SoftDevice)
- RAM: ~32KB (including BLE stack)
- Buffer Allocation: 512 bytes total (256B TX + 256B RX)

Real-Time Performance

- Sample Acquisition Latency: <100μs from DRDY to packet ready
- Transmission Latency: <5ms (BLE), <1ms (USB)
- Maximum Throughput: 2000 SPS × 4 channels × 27 bytes = ~216 KB/s
- Timer Resolution: 30.5μs (32.768 kHz RTC)
- Button Debounce: 50ms hardware debouncing for reliable input

Communication Specifications

- BLE Range: 10+ meters (Class 2)
- USB Standard: USB 2.0 Full Speed (12 Mbps)
- Packet Loss: <0.01% under normal conditions
- Connection Recovery: automatic, <3 seconds

IV. Host-Side API Overview

4.1 Core features

- Graphical User Interface (GUI) for all core features (Section 5.4)
- Virtual Daisy Chaining[5] device management: simultaneous connections to multiple BCI units
  - The *BCIManager* class coordinates data acquisition and synchronizes samples
- Remote Firmware Configuration: The following parameters can be set dynamically through the API
  - Sample rates (250/500/1000/2000 SPS)
  - Channel gains for each channel (1x-24x)
  - Device Naming for multi-device setups
  - Input Mode: Switching between monopolar and differential electrode configurations
  - Event Synchronization across all connected devices
- Operation Modes

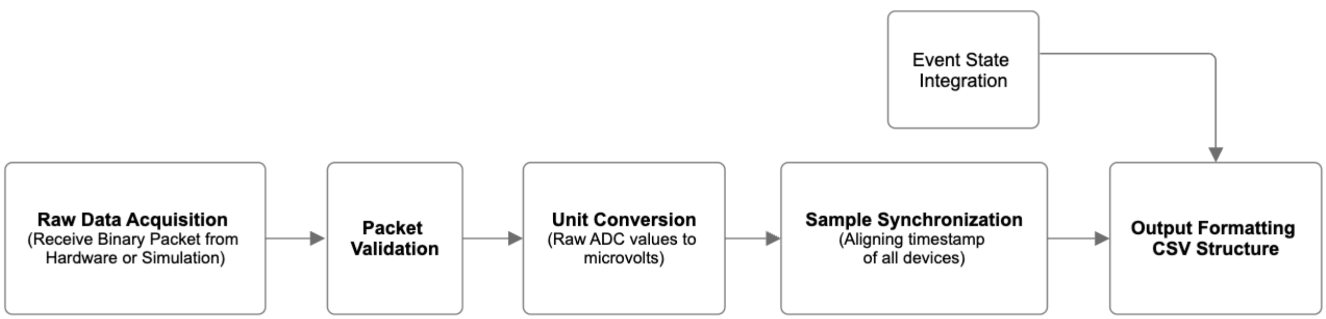
Record Mode	Lossless data logging to CSV files with this structure: sample_id, timestamp_us, [device_name]_ch0_uV, [device_name]_ch1_uV, ..., [event_name_0], [event_name_1], ... • 4 channels • Up to 8 boolean events, set by the user upon launching the GUI
Livestream Mode	• Toggle Console Output • Toggle Real-time Graphical Visualization • Toggle Quiet Mode (background streaming) • On-demand access to the latest sample data through API callback

[5] Removed Daisy Chaining hardware setup from the PCB

Passive Mode	Default mode. Active logging of device connectivity and configuration states.

4.2 Technical Features

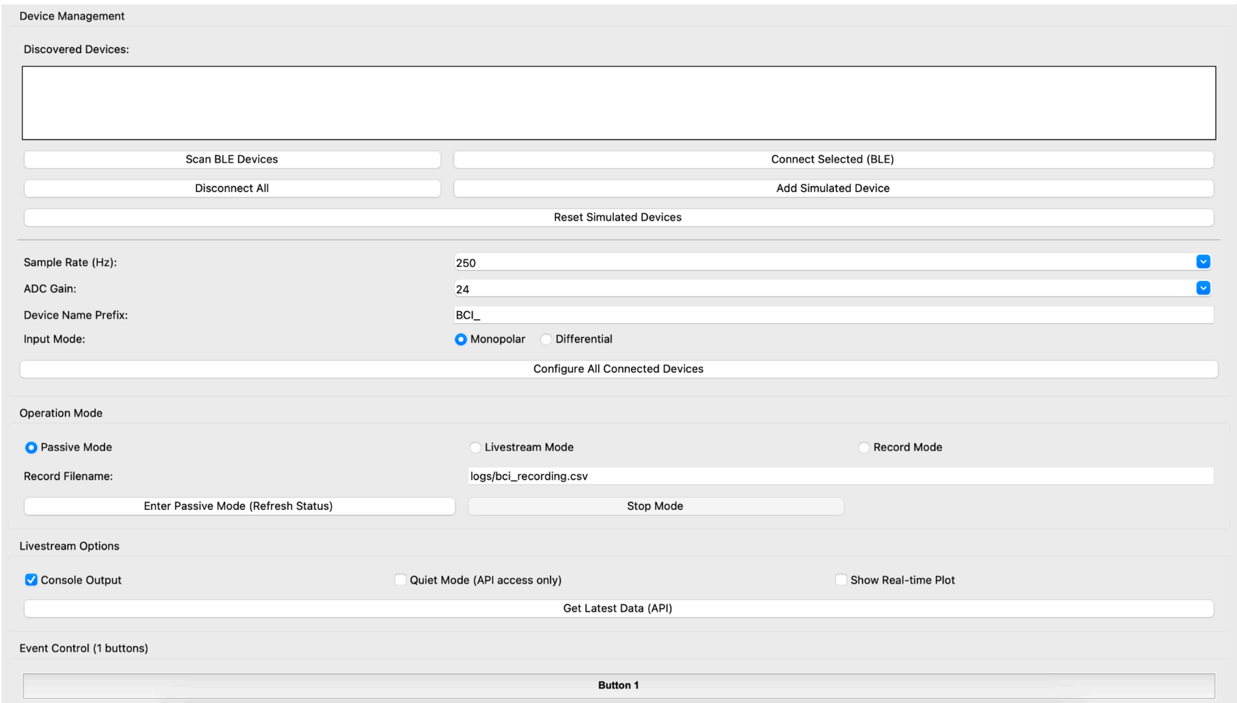
- Thread-safe Design: Multi-threaded architecture
  - Separate Data Acquisition Threads
  - Processing: dedicated threads for packet parsing and unit conversion
  - Queue-based communication: Thread-safe data flow (Python's *queue* module)
- Error handling and connection recovery
- USB support via the *pySerial* library
- BLE support via the *Bleak* library
- Data Processing Pipeline:



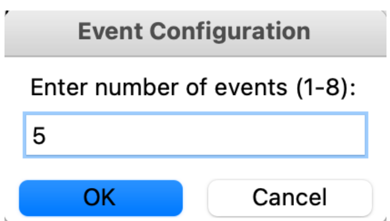
4.3 Simulation System

- API and GUI testing without physical hardware
- Realistic EEG data generation
  - Generates alpha (8-12Hz), beta (20Hz), and gamma (40Hz) frequency components to simulate multi-band neural signals
  - Includes 1/f pink noise and random artifacts (0.5% occurrence rate)
  - Each channel has unique phase offsets and gain variations
  - Timestamp matches theoretical hardware setup

4.4 GUI Application

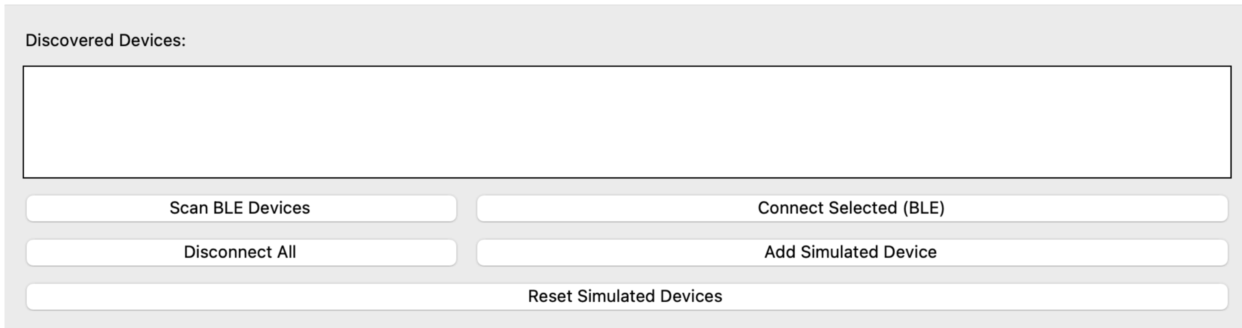


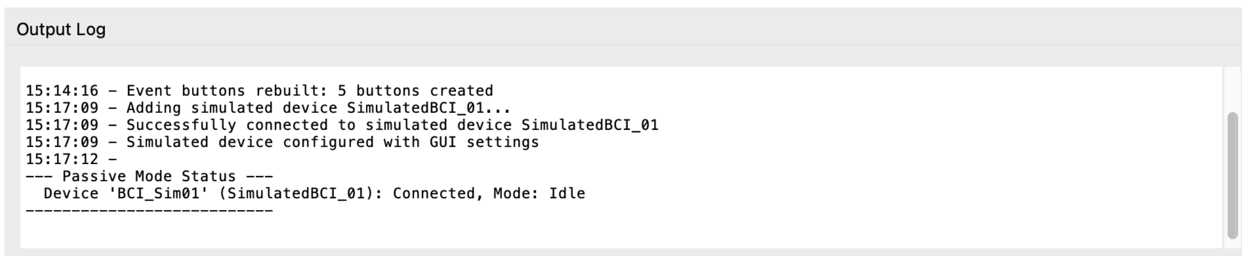
- Upon launch, users are prompted to enter the number of boolean events



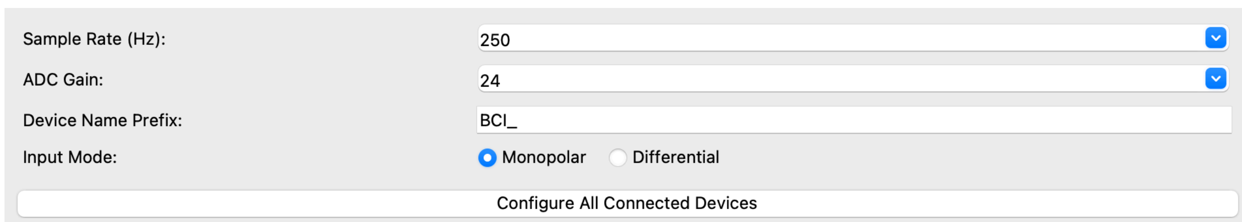
- The GUI will update its button amounts accordingly

- Users can scan, connect, and disconnect BLE devices. They can also add new or reset simulated devices. Device status is shown in the dialog box.

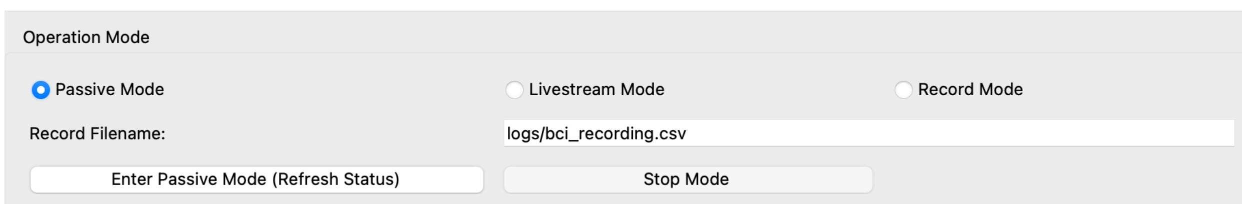


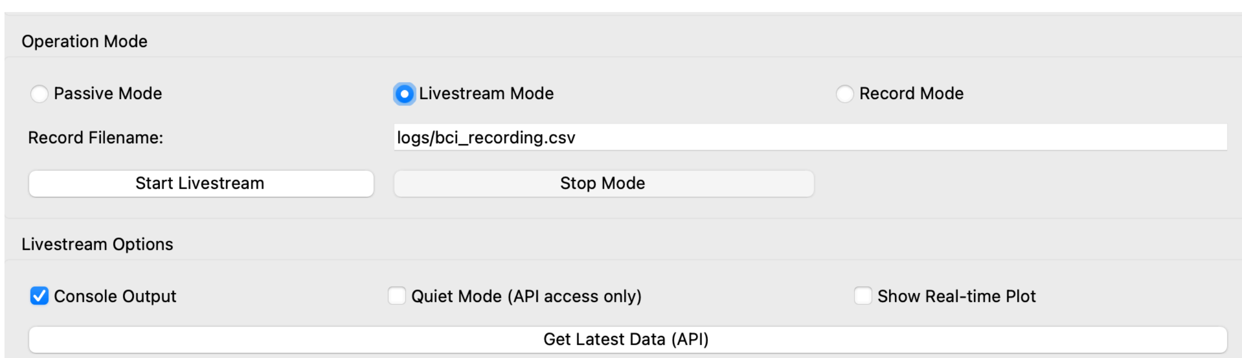


- Next, users can dynamically set the firmware parameters



- Mode selection





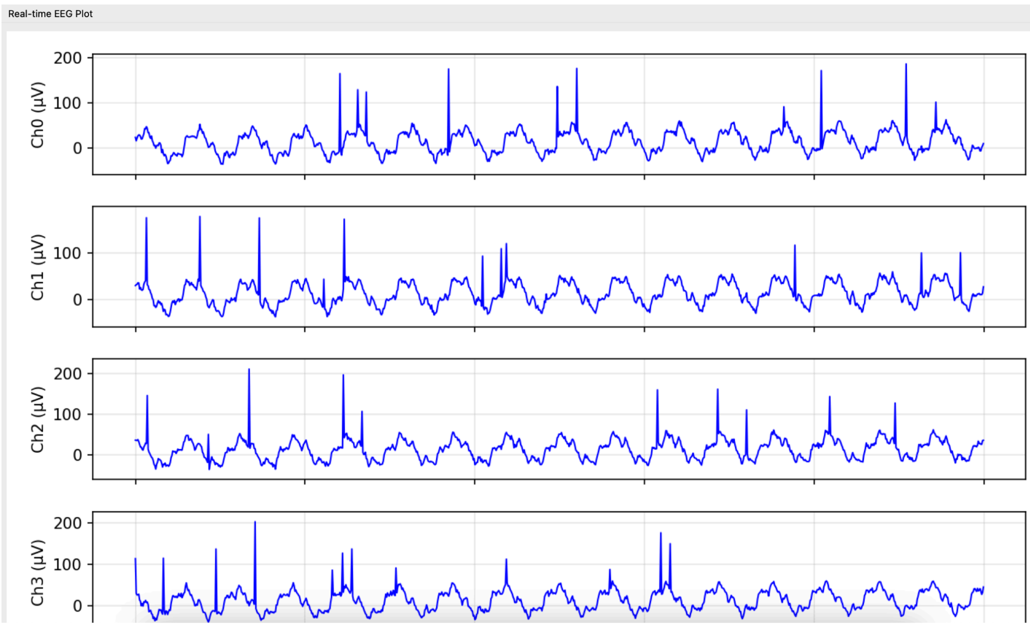
- Livestream mode:

  If the user enables console output, the data will appear in the log.

Output Log

```
15:20:07 - ID:1181, TS:175327438, Ch0:3.58µV, Ch1:11.65µV, Ch2:20.89µV, Ch3:33.76µV
15:20:07 - ID:1182, TS:175330052, Ch0:-0.65µV, Ch1:6.44µV, Ch2:22.55µV, Ch3:40.93µV
15:20:07 - ID:1183, TS:175333726, Ch0:-1.96µV, Ch1:9.06µV, Ch2:20.80µV, Ch3:41.34µV
15:20:07 - ID:1184, TS:175336300, Ch0:-1.63µV, Ch1:6.49µV, Ch2:21.99µV, Ch3:37.29µV
15:20:07 - ID:1185, TS:175339040, Ch0:-0.34µV, Ch1:8.44µV, Ch2:19.38µV, Ch3:34.79µV
15:20:07 - ID:1186, TS:175341571, Ch0:-1.45µV, Ch1:11.45µV, Ch2:23.03µV, Ch3:41.88µV
15:20:07 - ID:1187, TS:175344097, Ch0:1.09µV, Ch1:11.44µV, Ch2:33.42µV, Ch3:35.69µV
15:20:07 - ID:1188, TS:175346622, Ch0:-0.33µV, Ch1:12.70µV, Ch2:27.75µV, Ch3:37.57µV
15:20:07 - ID:1189, TS:175352260, Ch0:-7.04µV, Ch1:9.84µV, Ch2:27.69µV, Ch3:41.32µV
15:20:07 - ID:1190, TS:175356128, Ch0:-6.88µV, Ch1:10.34µV, Ch2:25.79µV, Ch3:35.47µV
```

If the user enables real-time plot, the graphs will appear under the log



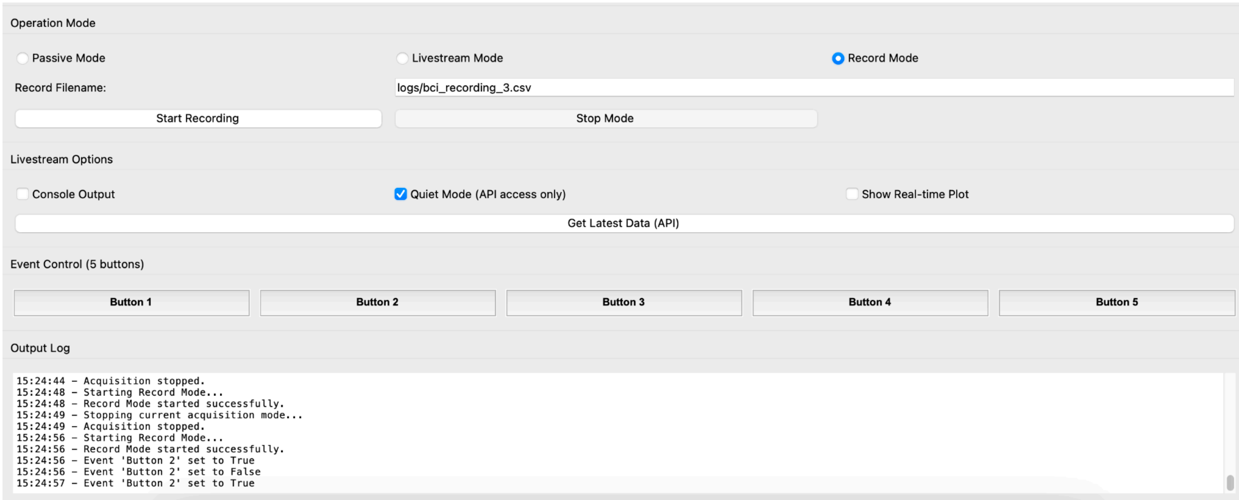
Enabling Quiet mode means the console will not show the continuous data, but the latest data can be called by pressing “Get Latest Data (API)” when needed.

Output Log

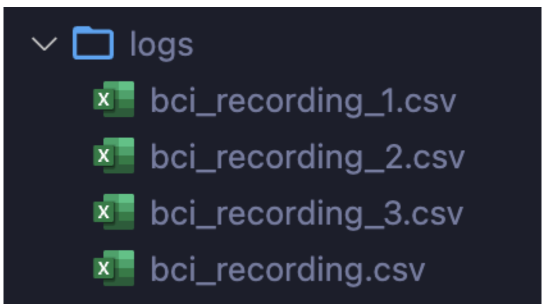
```
15:22:11 - Starting Livestream Mode...
15:22:11 - Livestream Mode started successfully.
15:22:15 - Latest API Data: sample_id:817, timestamp_us:304062410, BCI_01_ch0_uV:18.39, BCI_01_ch1_uV:28.25, BCI_01_ch2_uV:38.05, BCI_01_ch3_uV:49.79, Button 1:0, Button 2:0, Button 3:0, Button 4:0, Button 5:0
15:22:16 - Latest API Data: sample_id:881, timestamp_us:304232641, BCI_01_ch0_uV:28.74, BCI_01_ch1_uV:41.34, BCI_01_ch2_uV:48.24, BCI_01_ch3_uV:56.30, Button 1:0, Button 2:0, Button 3:0, Button 4:0, Button 5:0
15:22:17 - Latest API Data: sample_id:1013, timestamp_us:304580681, BCI_01_ch0_uV:40.40, BCI_01_ch1_uV:48.96, BCI_01_ch2_uV:44.75, BCI_01_ch3_uV:44.85, Button 1:0, Button 2:0, Button 3:0, Button 4:0, Button 5:0
15:22:20 - Stopping current acquisition mode...
15:22:25 - Acquisition stopped.
```

- Record Mode:

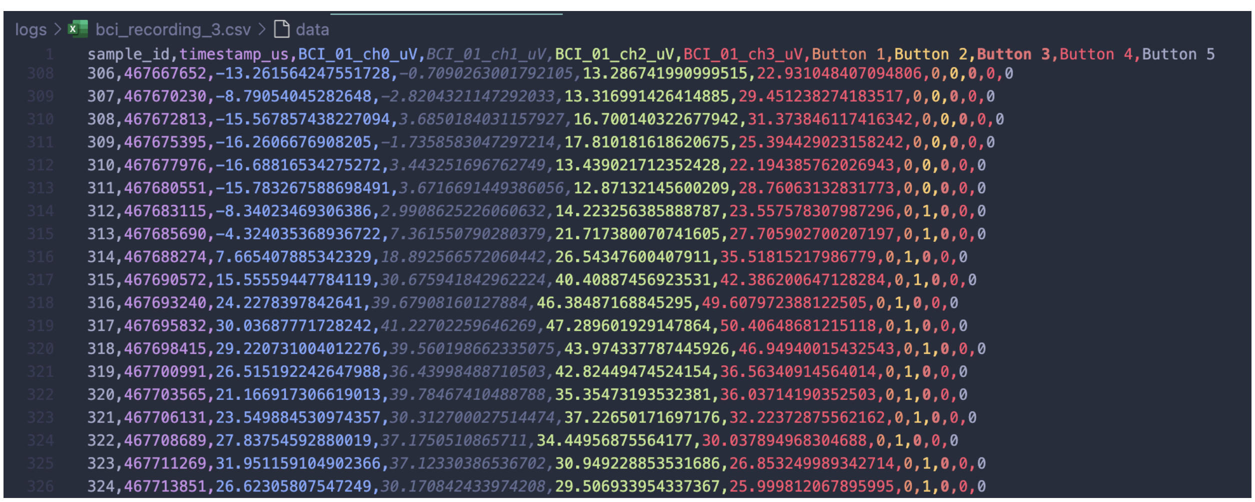
  The file name can be edited when record mode is activated.



The CSV files are saved in the logs folder.



Pressing Button 2 on the GUI resulted in the Button 2 column logging corresponding true values.


logs > bci_recording_3.csv > data

```
sample_id,timestamp_us,BCI_01_ch0_uV,BCI_01_ch1_uV,BCI_01_ch2_uV,BCI_01_ch3_uV,Button 1,Button 2,Button 3,Button 4,Button 5
306,467667652,-13.261564247551728,-0.7090263001792105,13.286741990999515,22.931048407094806,0,0,0,0,0
307,467670230,-8.79054045282648,-2.8204321147292033,13.316991426414885,29.451238274183517,0,0,0,0,0
308,467672813,-15.567857438227094,3.6850184031157927,16.700140322677942,31.373846117416342,0,0,0,0,0
309,467675395,-16.2606676908205,-1.7358583047297214,17.810181618620675,25.394429023158242,0,0,0,0,0
310,467677976,-16.68816534275272,3.443251696762749,13.439021712352428,22.194385762026943,0,0,0,0,0
311,467680551,-15.783267588698491,3.6716691449386056,12.87132145600209,28.76063132831773,0,0,0,0,0
312,467683115,-8.34023469306386,2.9908625226060632,14.223256385888787,23.557578307987296,0,1,0,0,0
313,467685690,-4.324035368936722,7.361550790280379,21.717380070741605,27.705902700207197,0,1,0,0,0
314,467688274,7.665407885342329,18.892566572060442,26.54347600407911,35.51815217986779,0,1,0,0,0
315,467690572,15.55559447784119,30.675941842962224,40.40887456923531,42.386200647128284,0,1,0,0,0
316,467693240,24.2278397842641,39.67908160127884,46.38487168845295,49.607972388122505,0,1,0,0,0
317,467695832,30.03687771728242,41.22702259646269,47.289601929147864,50.40648681215118,0,1,0,0,0
318,467698415,29.220731004012276,39.560198662335075,43.974337787445926,46.94940015432543,0,1,0,0,0
319,467700991,26.515192242647988,36.43998488710503,42.82449474524154,36.56340914564014,0,1,0,0,0
320,467703565,21.166917306619013,39.78467410488788,35.35473193532381,36.03714190352503,0,1,0,0,0
321,467706131,23.549884530974357,30.312700027514474,37.22650171697176,32.22372875562162,0,1,0,0,0
322,467708689,27.83754592880019,37.1750510865711,34.44956875564177,30.037894968304688,0,1,0,0,0
323,467711269,31.951159104902366,37.12330386536702,30.949228853531686,26.853249989342714,0,1,0,0,0
324,467713851,26.62305807547249,30.170842433974208,29.506933954337367,25.999812067895995,0,1,0,0,0
```

V. Deployment Notes

5.1 Firmware Flashing

Hardware Requirements

- Programmer: J-Link compatible debugger **OR** nRF52840-DK
- Debug Interface: Serial Wire Debug (SWD) pins and connector (Already included on the PCB)

Software Requirements

- J-Link Software Pack (latest version)
- nRF Command Line Tools
- Nordic SDK 17.1.0 and its SoftDevice (S140)
- SEGGER Embedded Studio

Flashing Procedure

Method 1 (nRF Command Line)

1. Connect Hardware
    - Connect J-Link to SWD pins on your PCB (VCC, SWDIO, SWDCLK, GND)
    - Ensure proper power supply (3.3V to VCC pin or via USB-C)
2. In the nRF Command Line, erase and flash SoftDevice

    nrfjprog --family nRF52 --chiperase

    nrfjprog --family nRF52 --program s140_nrf52_7.[**x.x**]_softdevice.hex
3. In the nRF Command Line, flash application firmware, reset, and start

    nrfjprog --family nRF52 --program **[application_name]**.hex --sectorerase

    nrfjprog --family nRF52 --reset

Method 2 (Segger Embedded Studio)

1. Open the project in SEGGER Embedded Studio
2. Build the project (F7)
3. Target → Connect J-Link
4. Target → Erase All
5. Target → Download project (Ctrl+T)

5.2 Running Host API & GUI

Hardware Requirements

- Windows 10/11, macOS 10.14+, or Linux (Ubuntu 18.04+)
- USB Port or Bluetooth Enabled

Software Dependencies

- Python: Version 3.7 or higher
- Install Required Python Packages:

```
pip install pyserial bleak matplotlib numpy
```

Installation

1. Clone or Download the API

```
git clone <repository-url>
cd bci-pcb-api
```

2. Install Dependencies

```
pip install -r requirements.txt
```

OR

```
pip install pyserial      # USB serial communication
pip install bleak         # Bluetooth Low Energy (BLE)
pip install matplotlib    # Real-time plotting
pip install numpy         # Numerical processing
```

Running the API

```
python bci_gui_app.py
```

5.3 Troubleshooting

Programming Issues

<u>Error Upon Building in Segger Embedded Studio:</u>

- Open flash_placement.xml with a text editor

- There are two instances of “rodata.” Near these instances is the word “size”. Change both “size” to “align.” Save the file.

```
rodata
<!DOCTYPE Linker_Placement_File>
<Root name="Flash Section Placement">
  <MemorySegment name="FLASH1" start="$(FLASH_PH_START)" size="$(FLASH_PH_SIZE)">
    <ProgramSection load="no" name=".reserved_flash" start="$(FLASH_PH_START)" size="$(FLASH_START)-$(FLASH_PH_START)" />
    <ProgramSection alignment="0x100" load="Yes" name=".vectors" start="$(FLASH_START)" />
    <ProgramSection alignment="4" load="Yes" name=".init" />
    <ProgramSection alignment="4" load="Yes" name=".init_rodata" />
    <ProgramSection alignment="4" load="Yes" name=".text" size="0x4" />
    <ProgramSection alignment="4" keep="Yes" load="Yes" name=".sdh_soc_observers" inputsections="*(SORT(.sdh_soc_observers*))"
address_symbol="__start_sdh_soc_observers" end_symbol="__stop_sdh_soc_observers" />
```

```
    <ProgramSection alignment="0x100" load="Yes" name=".vectors" start="$(FLASH
    <ProgramSection alignment="4" load="Yes" name=".init" />
    <ProgramSection alignment="4" load="Yes" name=".init_rodata" />
    <ProgramSection alignment="4" load="Yes" name=".text" align="0x4" />
    <ProgramSection alignment="4" keep="Yes" load="Yes" name=".sdh_soc_observer
address_symbol="__start_sdh_soc_observers" end_symbol="__stop_sdh_soc_observers
    <ProgramSection alignment="4" keep="Yes" load="Yes" name=".pwr_mgmt_data" i
```

```
  <ProgramSection alignment="4" keep="Yes" load="Yes" name=".log_filter_data"  inputsections="*
nin=".log_filter_data_run"/>
  <ProgramSection alignment="4" load="Yes" name=".dtors" />
  <ProgramSection alignment="4" load="Yes" name=".ctors" />
  <ProgramSection alignment="4" load="Yes" name=".rodata" align="0x4" />
  <ProgramSection alignment="4" load="Yes" name=".ARM.exidx" address_symbol="__exidx_start" end
  <ProgramSection alignment="4" load="Yes" runin=".fast_run" name=".fast" />
  <ProgramSection alignment="4" load="Yes" runin=".data_run" name=".data" />
  <ProgramSection alignment="4" load="Yes" runin=".tdata_run" name=".tdata" />
</MemorySegment>
<MemorySegment name="RAM1" start="$(RAM_PH_START)" size="$(RAM_PH_SIZE)">
  <ProgramSection load="no" name=".reserved_ram" start="$(RAM_PH_START)" size="$(RAM_START)-$(R
```

“Could not connect to target”

- Check SWD Connections
- Verify power (3.3V on VCC pin)
- Try to lower SWD Clock speed

  nrfjprog --clockspeed 1000

"Flash operation failed"

- Check if device is in debug interface lock
- Perform chip erase

  nrfjprog --family nRF52 --recover

- Verify J-Link firmware is up to date

"SoftDevice not found"

- Ensure SoftDevice is flashed before application
- Check SoftDevice version compatibility
- Verify flash memory layout in linker script

Runtime Issues

Device not advertising

- Check button functionality (short press should trigger advertising)
- Verify BLE stack initialization in logs
- Check if device name is properly set

No USB enumeration

- Verify USB power detection circuit
- Check CDC ACM class initialization
- Test with different USB cables/ports

No EEG data output

- Check ADS1299 SPI communication
- Verify DRDY interrupt configuration
- Check mode setting (must be Livestream or Record)

"No module named 'bleak'"

- If on Linux/macOS, try:

  pip3 install bleak

Serial Port Access Denied (Linux):

```
sudo usermod -a -G dialout $USER
# Then logout and login again
```

Log Monitoring

- Open J-link RTT Viewer Software
- Ensure NRF_LOG_BACKEND_RTT_ENABLED is set to 1 in sdk_config.h
- File → Connect
- In Specify Target Device, choose NRF52840_XXAA
- Keep the rest of the settings and click **OK**

5.4 Testing

Functionality Test

- Power-on self-test (POST)
- SPI communication with ADS1299 (check logs for initialization messages)
- BLE advertising and connection
- USB enumeration and communication
- Button response (short/long press)
- LED indicator functionality
- Mode switching via commands
- Event flag processing

Performance Tests

- Sample rate accuracy at 250/500/1000/2000 SPS
- Data integrity over continuous operation (1 hour)
- BLE connection stability test
- USB data throughput verification
- Power consumption measurement per mode

VI. Limitations and Future Improvements

6.1 Current Limitations

Hardware Constraints

- 4-Channel Limit and Monopolar Reference leads to less spatial resolution
- Battery Life is estimated to be 8–12 hours at 1000 SPS with BLE. Charging time may interrupt longer experiments
- Limited Range
- Relatively bulky Form Factor and Cabling Issues

Firmware & API Limitations

- Lack of real-time filtering or artifact rejection.
- Small buffer (256 bytes)
- ASCII Command Protocol lacks error handling
- Lack of encryption and multi-host support
- Advanced settings require firmware rebuilds
- Host platforms limited

6.2 Future Improvements

Hardware Improvements

- Expand Channel Capacity
    - Use ADS1299-8 for 8-channel EEG acquisition
    - Configurable differential input pairs to improve spatial resolution
- Improve Power Management
    - Extended battery life through ultra-low-power sleep modes, adaptive sampling rates based on signal activity, and smarter power-efficient BLE connection intervals
    - Wireless charging or energy harvesting
- Wearability
    - Reduced form factor
    - Flexible or skin-safe PCB
    - Direct electrode interface without cables

Firmware & API Improvements

- On-device, Real-time digital filters (high-pass, low-pass, notch)
- Artifact detection and flagging (blinking, motion, etc.)
- Machine Learning Integration for noise reduction, signal enhancement, sample rate adjustment, and power management
- Encrypted data transmission
- Predictive analytics for medical detection and classification
- Cross-platform GUI: Mobile application and web browsers
- Cloud Storage for secure data backup and sharing

VII. Appendix

Schematic, PCB, and code files can be found at the following github repository:

https://github.com/mandyliao66/BCI_Documents/tree/main