Kelly Jantzen

Western Washington Unversity |

easy\_bci

a matlab based application for classroom exploring of the basics of brain computer interfaces

easy\_bci

The easy\_bci is an open source, Matlab based application that allows students build custom applications that collect, analyze and display EEG data in real time. The goal is to provide a platform for students to gain experience working with all components of a simple brain computer interfaces within minimal programming experience. Because the goal was for students to gain some programming exposure, this suite is not designed to be a complete user interface. Instead it packages analysis routines into simple callable objects that allow students to construct a processing stream from a set of existing modules.

This manual provides an overview of the components of easy\_bci and instruction on how to get started as well as examples of how to create data handlers and analysis extensions.

overview

The easy\_bci application is comprised of two main components.

1. The hardware device for reading EEG.
2. The set of MATLAB routines for reading, analyzing and displaying the EEG data.

# The hardware

The easy\_bci was designed as part of an effort to create a very low-cost EEG machine suitable for classroom demonstrations and laboratory classes focused on neurophysiological measurement, particularly EEG and ERP research. With that in mind the easy\_bci is based on a modified version of the Heart & Brain Spiker Box (HBSpiker) from Back-yard-brains. This 1 channel EEG device was approximately $150 US at the time we purchased them and approximately $190 US at the time this manual was produced. The device was altered in the following ways to make it more suitable for simple student led ERP experiments.

1. Custom firmware was written to allow the device to operate in either a continuous collection mode or a single trial collection mode. The single trial mode is suitable for collecting short intervals of EEG that are time locked to an external event marker. This mode allows for easy real-time calculation and display of ERP’s without the need an additional low level ring buffer for handling trial onset detection and pre-trial data storage. In both Continuous and Single Trial modes, the firmware also samples any digital event markers communicated over the accessory pins of the HBSpiker and includes them as a second channel in the data packet. This firmware is available in the Arduino subfolder.
2. A UART bridge (using a SEEED RP2040 microcontroller) was added to allow event markers to be generated from any device or software that can write signals to a virtual serial port (e.g. to a USB port). The firmware for this RP2040 is available in the Arduino Code subfolder.

# The software

The software is all written in MATLAB and freely available. There are 4 main components of the easy\_bci application. The first is the main interface

The easy\_bci interfaces with the customized *BNS\_HBSpiker* via a BNS\_HBSpiker object written in MATLAB. This simple object (found in the Devices subfolder) handles interactions with the hardware including setting up serial communication, putting the device in the desired collection mode, and reading/decoding available data packets. It also handles the transfer of new data packets to the user defined data handler.

In theory, the user could write their own “device driver” to interface with whatever device they have or build. For example, I have built my own single channel EEG machine and use the same software tools (and its own device driver) for data collection and analysis.

But because theGraphical user interface

Description automatically generated interface to this device is controlled by a separate callable Matlab object, any similar driver could be created for any device. s The easy\_bci current works with a 1 channel EEG machine that is well suited for hobbyists, teachers and anyone interested in recording signals from their brain. Most low cost EEG devices provide access to frequency based measures of brain function including attention and relaxation. Although the ERPmini provides access to the same EEG signals on which that information is based, it’s real benefit is that it also allows user to access the brain signals that associated with the processing of very specific information such as reading a particular word or pushing a button. The signals tend to be very small and require averaging over many examples or trials to generate what is known as an event related potential or ERP.

An ERP is the averaged signal of the brain response to an external stimulus (exogenous response) or internal cognitive process (endogenous response). A key component of ERPs is that they are time locked to a known event of interest such as a stimulus onset or the timing of a motor response. In laboratory settings, EEG is collected continuously and event markers that signal the occurrence of potential time locking events are collected with the EEG. These event markers are used for the offline calculation and analysis of ERPs. However, there are applications, for which the user needs to calculate the ERP from the continuous data stream in real time. For example, the P300 speller requires calculating the P300 ERP on each trial as the user mentally selects a desired target from among options presented simultaneously on a screen. This tools and technical demands to accomplish this are make this kind of processing unavailable to most hobbyists, educators, students and others who are interested in creating fun applications using their own brain waves.

The ERPmini was designed specifically to remove technical barriers and allow users easy access to EEG signals more compatible with the online calculation of ERPs. Similar to other low-cost EEG options, the ERPmini can stream EEG continuously to a host computer via a USB connection. However, what makes the ERPmini unique is that it can also pass the user individual trials of data that are associated with a specific cognitive or motor event. To accomplish this, the ERPmini accepts external event markers and uses those to collect a small range of data (the default is -.125 s to +.75 s) around the onset of the signaling event. Rather than attempting to extract data from a continuous data stream, the simply reads the entire single trial as a single data packet. Each new event will generate a new trial and the user simply needs to average them together to create the ERP.

# Operating Modes

## Continuous Mode

In continuous mode, data are streamed continuously to the host PC at 512 Hz (see details below). Continuous mode is useful for viewing and analyzing the EEG in real time. The ERPmini continuously fills the serial port buffer with data as it is collected. It does not buffer data locally or monitor for serial buffer over run. It is the responsibility of the host computer to ensure that data can be read from the buffer at a rate that does not cause significant lags between when the EEG was sample and when it was read by the host computer.

## Single Trial Mode

In single trial mode, the ERP mini read EEG data and stores it internally until it receives a an event marker over the trigger (TRIG) port. It then collects the remaining post stimulus samples and passes the entire trial to the host computer. ERPmini reads only the first two bits of the event marker, allowing it to distinguish between 3 different trial types. Trigger signals that mark different events can be generated by sending a decimal 1, 2 or three over the serial port of the stimulus/response computer.

# Quick Start

1. Apply EEG electrodes (Active, Reference, Ground) to the scalp, for recording EEG, or other areas of the skin for recording EMG, EKG, etc.
2. Plug your touch proof EEG electrode cables into the Active (ACT), Reference (REF) and Ground (GND) electrode input jacks.

## For Continuous Mode

1. Connect the ERPMini to the the USB port on your host computer.
2. Establish a 57600 BAUD serial connection between the host computer and the ERPmini.
3. \*Send a command string to the ERPmini to place it in continuous mode (see configuring the ERPmini). This step is optional
4. Start reading data (see the section below describing the data packet structure)

\*This step is optional if the ERPmini just powered up because it defaults to continuous mode. However, it will be necessary if the ERPmini was previously put into single trial mode.

## For Single Trial Mode

1. Connect the ERPMini to the USB port on your host computer.
2. Connect the stimulus computer to the USB C TRIG input on the ERPmini
3. Establish a 57600 BAUD serial connection between the host computer and ERPmini.
4. Establish a 9600 BAUD serial connection between the stimulus computer\* and the ERPMini.
5. Send a command string from the host computer to the ERPmini to place it in single trial mode (see configuring the ERPmini)
6. Send a digital 1, 2 or 3 to the ERPmini TRIG port from the serial port of your stimulus computer.
7. Read the single trial data packet from the serial buffer on the host computer (see section below describing the single trail packet structure.

\*By using different USB ports, the same computer can act as both the host and the stimulus computer.

The ERPmini LED output

The following table provides a reference for interpreting the signals from the colored LEDs on the top of the ERPmini

|  |  |  |
| --- | --- | --- |
| Color | Behavior | Meaning |
| White | Solid | Waiting to establish a valid serial connection the host computer |
| Red | Solid | Electrodes are not connected to a participant or not plugged into the ERPmini |
| Yellow | Solid | Poor electrode connection |
| Green | Solid | Good electrode connection |
| Yellow | Flashing | ERPmini was successfully placed into continuous mode |
| Green | Flashing | ERPmini was successfully placed into single trial mode |
| Red flashing | Flashing | An unrecognized or unsupported command was received. |

Configuring ERPmini

ERPmini can be configured by sending command strings from the serial port of the host PC. The following table describes the commands and their effect.

|  |  |  |
| --- | --- | --- |
| Command | Effect | Description |
| cm0 | Enables continuous mode | Immediately places ERPmini in continuous mode. |
| cm1 | Enables single trial mode | Immediately places ERPmini in single trial mode. |

# Continuous Mode

To configure ERPmini to operate in continuous mode, send the three character sting cm0 to the ERPmini from the host computer. When it is first powered on, ERPmini will default to continuous mode. Successfully configuring ERPmini in continuous mode will cause the yellow LED to flash briefly.

In continuous mode, raw values are streamed to the host PC at 512 Hz. Each raw value is represented as two consecutive unsigned 8-bit integers. Details of the data packet are provided in the Continuous Mode section of The ERPmini Data Packet portion of this document.

# Single Trial Mode

To configure ERPmini in single trial mode, send the three-character string cm1 to the ERPmini from the host computer. Successfully configuring ERPmini in single trial mode will cause the green LED on ERPmini to flash briefly.

In single trial mode, ERPmini collects and transmits a single trial of data when it receives a trigger signal over the trigger input port (TRIG). By default, the trial contains a 125 ms (64 data points) pre-trigger period and a 750 ms (384 data points) post-trigger period. The pre-trigger period contains the raw EEG values from the 125 ms prior to receiving the trigger and the post trigger period contains the raw EEG values from the 750 ms after receiving of the trigger. Details of the single trial data packet are provided in the Single Trial Mode section of The ERPmini Data Packet portion of this document.

The ERPmini Data Packet

# Continuous Mode

When in continuous mode, data are streamed to the host PC at 512 Hz. The EEG is collected with 12-bit resolution in the range -2048 to 2047. Before transmitting, 2048 is added to make the value positive. It is then split across two bytes which are transmitted sequentially. The first byte contains the higher 5 bits, and the second byte contains the lower 7 bits. To ensure the two bytes are read in the correct order, the most significant bite (MSB) on the first byte will always be 1, and the MSB on the second byte will always be zero.

The example shows how a raw value of 643 would be represented across the bytes. The value 2048 has been added to the raw value to eliminate signed integers so the value below is a decimal 2961 (12-bit binary: 1010 1000 0011).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| High Byte | | | | | | | | Low Byte | | | | | | | |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| signals high byte | unused | unused | Highest 5 bits | | | | | signals low byte | Lowest 7 bits | | | | | | |

To combine the bytes into a 16-bit integer, use a bit mask to select the first 5 bits of the high byte (0x1F or 31) and the first 7 bits of the lower byte (0x7F or 127), shift the bits of the high byte to the left by 1 bit and combining them with the low byte. Subtract 2048 to put the data back into the original range.

The following is an example using c/c++ syntax.

raw\_sample = ((byte1 & 0x1F) >> 1 | (byte2 & 0x7F) -2048

According to Neurosky, the following is used to covert raw data samples to volts.

v = raw\_sample \* (1.8/4096) / 2000;

The input voltage to the TGAM1 is apparently 1.8, the 12-bit data range of 4096 and a gain of 2000 is applied. Multiply this value by 1 million to convert to microvolts (which is probably a more appropriate scale).

# Single trial mode

When in single trial mode, data is transferred in individual packets that includes a header that describes the data, and a data payload that contains the raw data bytes.

The header bytes are described in the following table. Literal characters are enclosed by quotations. Underscores represent the space character (ascii character 32). Italic characters are place holders for single or multiple 8 bit values.

|  |  |  |  |
| --- | --- | --- | --- |
| Bytes | Content | Data type | Description |
| 1-11 | “trial\_onset” | char | Marks the onset of a data packet for a single trial |
| 12 | *evt* | uint8 | *evt* is an 8-bit integer in the range 1 to 3 representing the state of the two lowest bits read on the trigger input cable. |
| 13-14 | *sr* | uint16 | *sr* is 2 consecutive unsigned 8 -bit integers that give the sampling rate when combined into an unsigned 16-bit integer. Should always be 512. |
| 15-16 | *preBytes* | uint16 | *two consecutive (high, low) unsigned 8-bit integers that, when combined into a*n unsigned 16-bit integer give the total number of bytes in the pre-trigger period. The total number of data points in the pre-trigger period will be *preBytes* / 2. |
| 17-18 | *pstBytes*\_ | uint16 | *two consecutive (high, low) unsigned 8-bit integers that, when combined into a*n unsigned 16-bit integer give the total number of bytes in the post-trigger period. The total number of data points in the post-trigger period will be *pstBytes* / 2. |

The data payload begins at byte 19. The number of bytes in the payload is equal to *preBytes + pstBytes.* Raw values are stored in the payload as 2-byte pairs and are the high (first byte) and low (second byte) bytes of a signed (two’s complement) 16-bit integer in the range -2048 to 2047. The packet is terminated with a CR/LF which allows convenient reading of the entire packet without having to know its length ahead of time.The entire packet length is the header length plus the data payload length and the terminator length, or:

*19 + preBytes + pstBytes + 2*