EEGu2: An Embedded Device for Brain/Body Signal Acquisition and Processing

Shen Feng, Mian Tang, Fernando Quivira, Tim Dyson, Filip Cuckov, Gunar Schirner
Northeastern University, Boston, MA
University of Massachusetts Boston, Boston, MA
{sfeng,fquivira,miantang,tdyson,schirner}@ece.neu.edu
filip.cuckov@umb.edu

ABSTRACT

Brain/Body Computer Interface (BBCI) technology facilitates research in human cognition and assistive technologies. BBCI acquires and analyzes physiological signals from human body/brain such as electroencephalography (EEG) to observe human physiological states and potentially enable external control. BBCI devices require accurate data acquisition systems with sufficient dynamic range for various brain/body signals. Also, embedded processing is desirable for real-time interaction and flexible deployment. However, most off-the-shelf BBCI devices are very costly, e.g. g.USBamp at $15K and do not offer embedded processing. Hence, an open embedded device for BBCI acquisition and processing is needed to foster the BBCI research.

This paper proposes EEGu2 as a portable embedded BBCI device. Based on a BeagleBone Black (BBB), EEGu2 integrates a custom-designed cape including 2 PCBs: an acquisition board for 16-channel 24-bit acquisition up to 1KHz sampling frequency and a power board for wall charging and powering mobile operations. EEGu2 measurement shows a high acquisition accuracy with 25dB signal-to-noise ratio and 0.785μV peak-to-peak input referred noise. At maximum performance, the cape consumes 101.2 mW while BBB consumes 1850 mW. With two lithium batteries, EEGu2 operates independently 12 hours.

We demonstrate the flexibility and portability of EEGu2 in the context of Human-in-the-Loop Cyber-Physical Systems (HiLCPS) that augments human interaction with physical world through BBCI. The EEGu2 firmware is integrated into the *HiLCPS Framework* to enable the location transparent access via the MATLAB interface. EEGu2 empowers rapid embedded BBCI application deployment and we show the flexibility of EEGu2 with a BCI Speller application that acquires real-time EEG signals and infers the user spelling based on Steady State Visually Evoked Potential.

CCS Concepts

•Computer systems organization → Firmware; Embedded hardware; Embedded software;

Keywords

Data Acquisition System, Body/Brain-Computer Interfaces, EEG, Embedded Real-time, Hardware Abstraction, Automated Embedded Deployment, Domain-Specific Synthesis

1. INTRODUCTION

RSP'16, October 06-07, 2016, Pittsburgh, PA, USA

DOI: http://dx.doi.org/10.1145/2990299.2990304

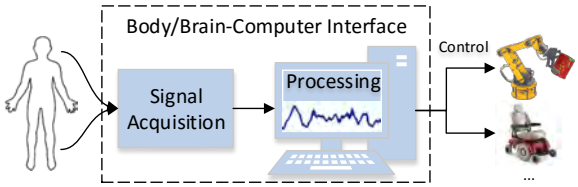


Figure 1: Body/Brain-Computer Interfaces

Body/Brain-Computer Interface (BBCI) is an emerging technology enabling a class of applications in which computers monitor human body and brain activity and interpret it as a feedback to human or an intent of control. The work in [1] uses BBCI to monitor heart health by measuring electrocardiography (ECG). The research in [2, 3] measures electroencephalography (EEG) to detect the user attention level in the treatment of Attention Deficit Hyperactivity Disorder (ADHD). BBCI also serves as an additional degree of freedom in control to augment the user interaction with the physical world. BBCI has been combined with assistive technologies in several contexts: operating prosthetic devices [4], brain-controlled robot [5], typing through virtual keyboards [6] and playing games [3].

Fig. 1 overviews an assistive application using BBCI. BBCI contains both Data Acquisition (DAQ) and processing. BBCI senses a variety of human physiological signals, including EEG, ECG, electromyography (EMG), electrooculography (EOG), which measure electrical signals emitted by the brain, the heart, skeleton muscles and eyes respectively. BBCI requires very high resolution and low noise DAQ for very small bio-signals such as EEG from 1 to 100μV. However, to measure EEG and larger bio-signal such as EMG (up to 10mV) at the same time, a sufficient dynamic range is needed.

DAQ outputs digitized bio-signals for further processing. BBCI algorithms identify different patterns of body and brain activity, each being associated with a human intention to control assistive devices such as a prosthetic arm or a wheelchair. Many current BBCI applications run on a personal computer, which is cumbersome for mobile operations. Hence, a portable (battery powered) embedded BBCI device with both acquisition and processing integrated is desired.

Developing an embedded BBCI application poses many challenges. One challenge is the lack of portable embedded BBCI devices with acquisition and sufficient processing capacity. Most commercial off-the-shelf products from companies like Emotive, g.Tec, Neurosky only provides DAQ and relies on an additional computer for signal processing. Moreover, the commercial DAQ products are mostly expensive, e.g. g.USBamp at around $15k. Alternatively, open source DAQ platform such as OpenBCI [7] is equipped with on-board micro-controller (50MHz), however, not

accessible to users. Another challenge is the complex algorithm development. Algorithm designers prototype BBCI application using tools such as MATLAB to benefit from its signal processing intensive algorithm design environment. However, the gap between MATLAB prototype and embedded deployment challenges algorithm designers with significant effort and embedded knowledge for manual conversion.

To address the above challenges, we propose EEGu2: an affordable and portable (battery powered) embedded system for both bio-signal acquisition and embedded processing. The contribution of this paper is three-fold:

1. **BBCI Acquisition and Processing Platform.** As a modular design, EEGu2 integrates a customized cape (PCB) stacking on top of BeagleBone Black (BBB) for 16-channel 24-bit bio-signal acquisition, embedded processing, and intelligent power management for mobile operation ($<$ 12hours).

2. **HW Type and Location Independent Access for Development.** EEGu2 firmware implements a canonical DAQ HW API and the underlying DAQ kernel driver. The firmware is further integrated into the *HiLCPS Framework* [8] to enable hardware and location transparent access via the MATLAB interface.

3. **Enabling Rapid BBCI Application Development.** Through domain-specific synthesis, the prototyped MATLAB BBCI algorithm can be automatically deployed onto embedded systems (e.g. EEGu2) with seamless integration of the firmware libraries.

EEGu2 empowers a rapid embedded BBCI application deployment with close-to-sensing processing. This paper demonstrates this flexibility of EEGu2 with a Brain-Computer Interface (BCI) Speller application in the context of HiLCPS that augments human interaction with the physical world.

This paper is structured as follows: Section 2 describes the EEGu2 architecture design, firmware and framework integration. Section 3 shows system design quality and demonstrates a BCI Speller application. Section 4 overviews related work of existing BBCI devices. Section 5 concludes the paper.

2. EEGU2: EMBEDDED BBCI

Fig. 2 depicts an EEGu2 with the upper cover removed. EEGu2 integrates a BeagleBone Black (BBB) as the base processing element. BBB Rev.C comprises a powerful AM3358 processor based on ARM Cortex-A8 (1GHz), 2x Programmable Real-time Unit (PRU), 512MB DDR3 (800MHz x 16bit) and 4GB on-board storage using eMMC. While offering low power embedded processing, BBB provides plenty of I/O and peripherals interfaces, such as SPI (interfacing acquisition board), USB (WiFi dongle), etc.

Figure 2: EEGu2

On top of BBB, EEGu2 contains a customized acquisition (DAQ) board equipped with 2x *ADS1299* chips [9], low noise, 24 bit, 2x8 channel analog front-end (see Section 2.1.1). EEGu2 is designed to be portable with two low-cost lithium-ion cell phone battery installed. A customized power board provides power management, including powering all circuitry and battery charging (see Section 2.1.2). The analog/digital power and signals are strictly isolated to avoid polluting bio-signals (see Section 2.1.3). A detachable 3D printed case provides the mechanical support for internal circuit boards (see Section 2.1.4). EEGu2 firmware implements the DAQ kernel driver for real-time acquisition and a software library providing canonical DAQ HW API (see Section 2.2). The firmware is further integrated into the *HiLCPS Framework* [10] as a DAQ backend, which enables transparent access from the MATLAB interface and automated embedded deployment from MATLAB application to EEGu2 (see Section 2.3).

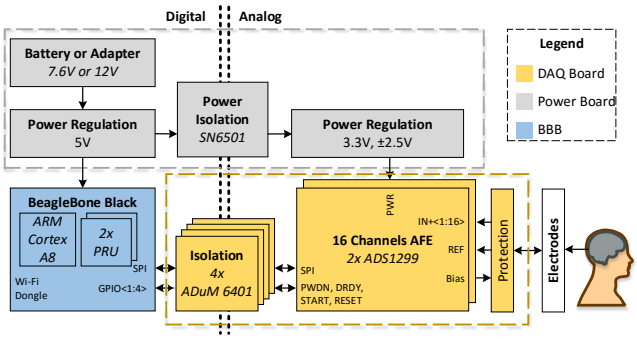


Figure 3: Architecture Overview

2.1 EEGu2 Architecture

Fig. 3 overviews the architecture of EEGu hardware design. The power board (in the upper dash box) provides digital power 5V to BBB (bottom left) and analog power to the DAQ board (in the bottom right dash box). The vertical double line separates the digital circuit island from the analog circuit island to avoid signal interference. Both power lines and signal lines across the analog/digital border are isolated. From bottom right human to bottom left BBB, body/brain signals go through the protection circuit and get digitized by 16-channel ADCs. The signal samples are sent to BBB over SPI through proper isolation. BBB can either run the BBCI application locally or stream samples over WiFi to a computer for processing.

2.1.1 DAQ Board

Fig. 4 and Fig. 6 depict the physical board and conceptual signal flow between components. Human bio-signals sensed from passive electrodes go through protection circuit to the central analog front end (AFE). All channel are configured single-ended. The protection circuit (IEC60601 standards [11]) limits the current flow to the patient. AFE comprises two *ADS1299* chips that support 16 channel acquisition. Each *ADS1299* contains programmable gain amplifier (up to 24x) and 24-bit delta-sigma analog-to-digital converters (ADC). This high resolution ADC is necessary for sensing EEG signal (around 10μV). The dynamic range is $\pm$4.5V at the gain of 1 and $\pm$187.5mV at the gain of 24. Even with the finest gain of 24, $\pm$187.5mV is a large dynamic range that allows mixed bio-signals (e.g. EEG, EMG) measurement and tolerates high noise level without ADC saturation. The sampling frequency of ADC

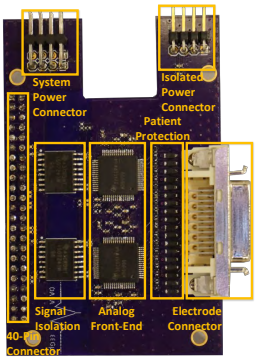


Figure 4: DAQ Cape

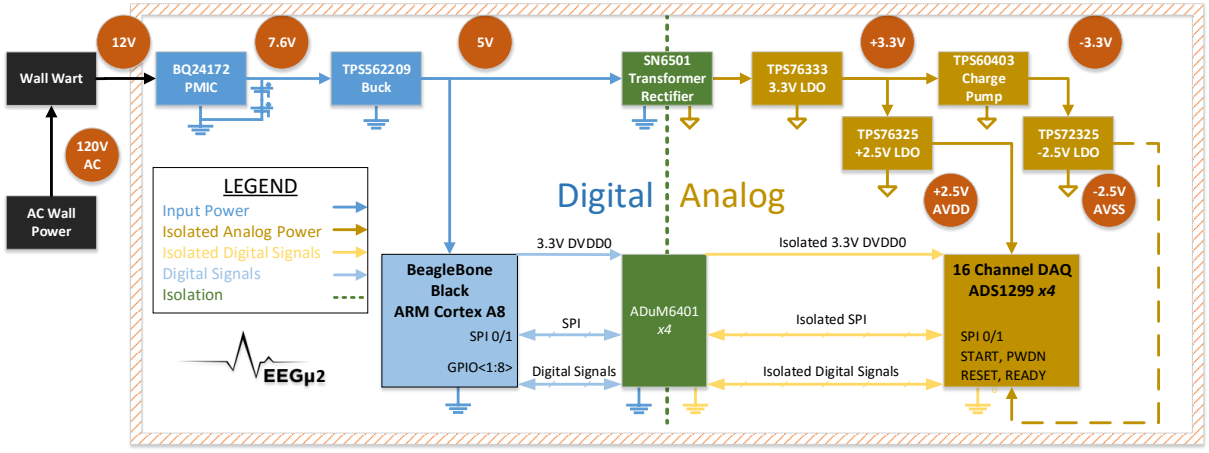


Figure 5: Power and Signal Isolation

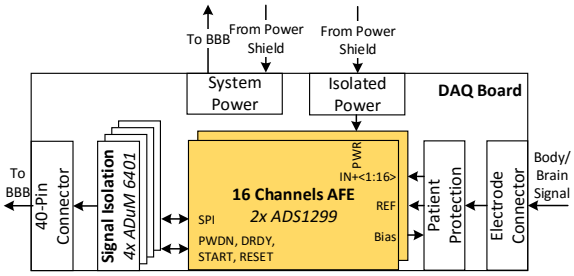


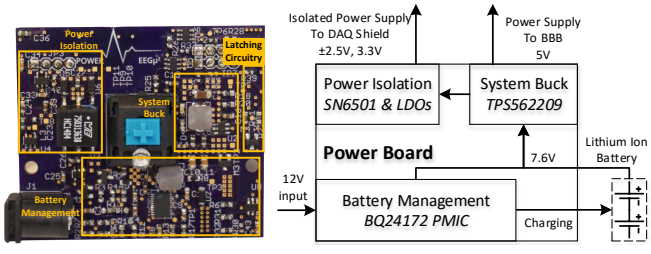
Figure 6: EEGu2 DAQ Cape Design

can be configured from 250 samples per second (SPS) to 16kSPS. Also, *ADS1299* is optimized for bio-signal sensing offering lead-off detection, bias sensing, and bias drive amplifier as the right leg drive [9].

The cape reads samples from AFE and streams it to BBB via SPI. To avoid introducing digital noise to ADC, all digital signals between BBB and AFE are isolated by *ADuM6401* isolators (each with 4 channels). Each *ADS1299* requires 2 *ADuM6401* for SPI and control lines respectively.

2.1.2 *Power Board*

Fig. 7 illustrates the EEGu2 power board that supports mobile operation with battery and wall charging. The power board contains a *BQ24172* power management IC (PMIC) that takes 12V input from an adapter connected to the wall outlet. The PMIC outputs the 7.6V rail and charges two lithium batteries (Samsung Galaxy Note 4 battery, 3200mAh, 3.8V) with LED status indicator. Those two batteries in series offer 7.6V. The system buck regulates 7.6V to 5V that goes to the PMIC on BBB and powers BBB. Several low-dropout (LDO) regulate 5V to 3.3V and ±2.5V to DAQ board. The power board is mounted on the DAQ board through both power



(a) Board (b) Design

Figure 7: Power Board

connectors.

2.1.3 *Power and Signal Isolation*

Fig. 5 illustrates the power and signal isolation to protect the ADC from digital noise. SPI and GPIOs are isolated when crossing the analog/digital border. Also, the 5V power rail from the system buck to the analog island is isolated by a transformer rectifier.

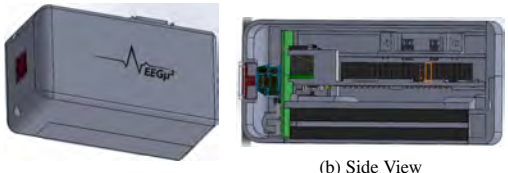
All power rails contain several levels of voltage regulation with the trade-off between noise and efficiency (listed in Table 1). A high efficient power buck drives both BBB and DAQ board. However, the buck must be isolated (via transformer rectifier) due to its high switching noise (65KHz) and ripples that could pollute very small sensed signals (e.g. EEG). LDOs further regulate the power with much less noise and therefore reside in analog island. The result of EEGu2 power consumption is reported in Section 3.1.1.

Component	Noise	Efficiency
System Buck TPS562209	high	93% @5V, 350mA
Transformer Rectifier SN6501	n/a	53% @10mA
LDO TPS76333	low	65.14% @7.6mA
LDO TPS76325	low	74.77% @7.6mA
Charge Pump TPS60403	n/a	95% @10mA

Table 1: Voltage Regulation Noise and Efficiency

2.1.4 *Mechanical Enclosure*

Fig. 8 shows the 3D printed acrylic enclosure that encapsulates the BBB, the cape and batteries. BBB is screwed on the enclosure for high mechanical stability. Fig. 8b shows the DAQ board horizontally stacking over BBB and the power board (front) vertically connected to the DAQ board for saving space. The batteries can be easily slid in and out for replacement. The enclosure selectively exposes a translucent power button, an Ethernet port, a power jack and a USB port for WiFi dongle. For user's convenience, the enclosure has a belt clip on the side for wearable BBCI applications.



(a) Front View (b) Side View

Figure 8: EEGu2 3D Printed Enclosure

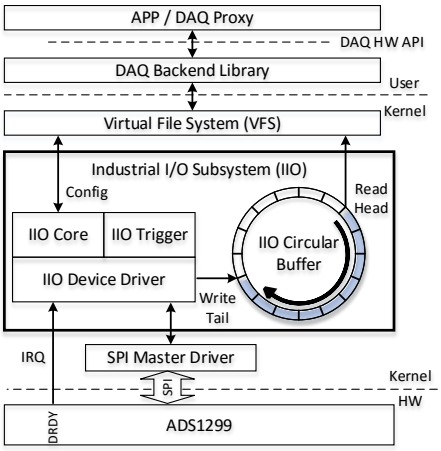


Figure 9: EEGu2 Firmware

2.2 EEGu2 DAQ Firmware

The EEGu2 firmware interfaces with the ADS1299 chips (for control and data) and provides an abstracted interface to the user program. An initial approach was based on a user-level driver, which however missed deadlines and lost samples (see analysis in Section 3.2.1)). Fig. 9 overviews our more robust version, with a kernel-level driver for SPI interaction and a virtual file system interface to the user level.

The DAQ kernel driver is based on Industrial I/O Subsystem (IIO) [12]. From bottom to top in Fig. 9, the data ready (DRDY) signal from *ADS1299* is an interrupt triggering two-level interrupt handling. The HW interrupt service routine (ISR) invokes the higher level IIO trigger ISR to read samples from the chip via SPI in a separate kernel thread. All channel samples are interleaved and pushed into the tail of IIO circular buffer.

The kernel driver interfaces with user programs through virtual file systems (VFS). A set of sysfs character device files provides access to the IIO buffer and driver configurations including trigger registration, channel gain, sampling frequency and buffer size. Listing 1 shows an example of driver usage. A file descriptor is created by opening the character device file of the IIO buffer. Then the program reads samples from the file descriptor (head of IIO buffer) upon the data availability using polling. Note the polling frequency here is not critical in the result of the driver data buffering.

2.3 HiLCPS Framework Integration

Our work in [8] proposes *HiLCPS Framework* to expedite the development of embedded HiLCPS applications that augment user interaction with the physical world via BBCI. Our framework provides unified access to similar hardware types independent of their location and domain-specific synthesis for automated embedded deployments.

Listing 1: EEGu2 Kernel Driver Interface

```
char filename[] = "/dev/iio:device0";
char buffer[BUF_SIZE]; //buffer to store samples
int samplePerCh=1; //samples per channel to read
/* Create input file descriptor */
int input_fd = open(filename, O_RDONLY);
struct pollfd pfd = {
    .fd = input_fd,
    .events = POLLIN,    // upon input
};  // error handling not shown for simplicity
poll(&pfd, 1, -1); // block til pfd data avail
// read samples of each channel
num = read(input_fd, buffer, samplePerCh * sizeof(int
    ));
...
close (input_fd);
```

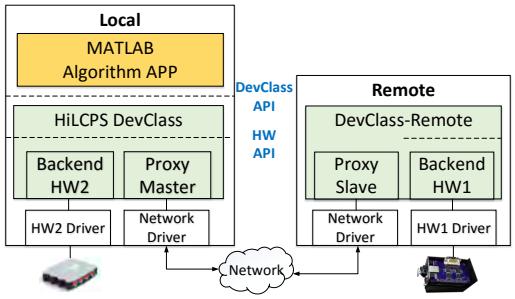


Figure 10: DevClass Hardware Type and Location Transparency

To interface with various hardware and connectivity, *HiLCPS Framework* groups similar HW types into classes, called *DevClass*. Each *DevClass* is a class of HW types that implement the same semantics of input and output. For example, DAQ is a DevClass of acquisition systems while EEGu2 is one HW type of DAQ DevClass. Fig. 10 depicts the structural composition of a *DevClass*. The *DevClass* implements a unified HiLCPS API in MATLAB. To support various HW types, a *DevClass* contains a set of HW-specific Backends that implement the canonical HW API by interfacing with actual HW drivers. The HiLCPS API enables a portable HiLCPS application design in that the application benefits from a consistent access to different hardware types within the class.

The *DevClass* also provides location transparent access to hardware regardless of its connectivity by using proxies. In comparison to the locally accessed HW2 (g.USBamp), the control of HW1 (EEGu2) is relayed through a pair of *proxy-master* and *proxy-slave* to *DevClass-Remote*. The *proxy-slave* runs a dedicated thread listening to *proxy-master* (e.g. over TCP) and notifying *DevClass-Remote* to execute the control by calling the Backend-HW1. On the backward path, the *DevClass-Remote* reads HW1 data and sends it back to *DevClass* through the proxies.

Listing 2 lists a snippet of DAQ HW API. The *init()* creates a DAQ object (struct instance) and returns its pointer value as the DAQ instance ID. Returning the object pointer value instead of the pointer itself allows the function being called without requiring the caller knowing the complex object definition. This design simplifies interfacing DAQ HW API with MATLAB through MEX (MATLAB executable for run-time access) and domain-specific synthesis (code generation for embedded deployment) during the framework integration. Other functions can refer to the DAQ object by casting the DAQ instance ID back. Given the amount of the samples requested per channel, *getSamples()* function returns a sample matrix, each column of which contains one channel samples.

In Fig. 11, the *HiLCPS Framework* realizes the following *DevClasses*: *Data Acquisition (DAQ)*, *Stimulus*, *Visualization*. The *DAQ DevClass* represents a class of DAQ devices sensing biosignal. Those DAQ devices share similar configurations such as

Listing 2: EEGu2 Firmware with DAQ HW API

```
// return initialized daq object pointer value
uint32_T init();
// store samplesPerCh*ActiveChNum sample matrix
// in preallocated data array (row major)
void getSamples(uint32_T pDAQ,uint32_T samplePerCh,
    uint8_T ActiveChNum, int32_T* data);
// turn on/off DAQ channel at Config 1/0
void configCh(uint32_T pDAQ, uint16_T ChNum, uint8_T
    Config);
// set DAQ sampling frequency
void setSamplingFreq(uint32_T pDAQ,uint16_T SampleFreq)
    ;
...
// clean up
void end(uint32_T pDAQ);
```

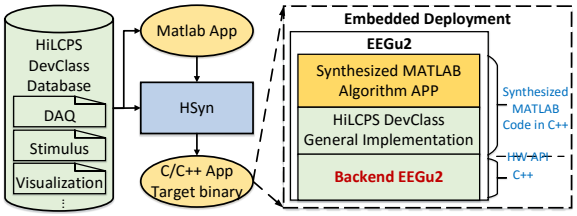


Figure 11: Rapid Embedded HiLCPS App Development

sampling rate and channel gain. EEGu2 is integrated into the framework as one supported DAQ device type in that EEGu2 firmware implements the DAQ HW API and serves as a DAQ *DevClasses* Backend.

With the framework integration, designers using EEGu2 benefit not only from the hardware and location transparent access in MATLAB but also automated embedded deployment enabled by domain-specific synthesis, as shown in Fig. 11. Algorithm designers first design and validate the algorithm model in MATLAB and rapidly expand it to the application with hardware-in-the-loop (using real sensors). The MATLAB application simply instantiates a *DevClass* (e.g. DAQ) to access a class of hardware (e.g. EEGu2) regardless of the hardware type and connectivity (e.g. TCP). Instead of worrying about hardware details (i.e. HW driver API), designers can focus directly on developing the algorithm. Finally, with domain-specific synthesis, the prototyped MATLAB application is automatically synthesized down to an embedded implementation in C/C++ and seamlessly interfaces with DAQ Backend library. More details regarding automated process of embedded deployment can be found in [10].

3. EXPERIMENTAL RESULTS

This section evaluates both the HW quality and firmware quality of EEGu2. We also demonstrate a use case of EEGu2 and the benefit of framework integration using a BCI Speller application.

3.1 HW Quality

The EEGu2 HW quality is evaluated in two aspects: 1) power consumption, 2) accuracy.

3.1.1 Power Consumption

As a portable, battery powered embedded system, the power consumption of EEGu2 is critical. Table 2 lists the measured power consumption of EEGu2 components. When system is idle (DAQ board idle, BBB idle), EEGu2 consumes 1219.5mW in which the cape consumes only 69.5mW. When running the DAQ driver (DAQ board busy) and *DevClass-Remote* for data streaming to PC (BBB busy), EEGu2 consumes 1951.2mW in total, that being 731.7mW more in comparison to the idle state. BBB contributes 1850mW, the majority of the overall power consumption due to embedded acquisition and WiFi communication. The cape consumes only 101.2mW when operating signal acquisition.

EEGu2 is equipped with two Samsung Galaxy Note 3 lithium ion battery, each with 3200mAh at 3.8V. This battery capacity translates to $(2 * 3200mAh * 3.8V)/1951.2mW = 12.46hour$ battery life time.

Component	Idle Power (mW)	Busy Power (mW)
Cape	69.5	101.2
BeagleBone Black	1150	1850
Total	1219.5	1951.2

Table 2: EEGu2 Power Consumption

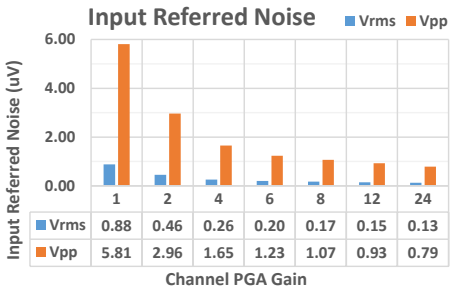


Figure 12: Input Referred Noise over all Gain

3.1.2 Accuracy

We evaluate accuracy as signal-to-noise ratio (SNR) and input-referred noise (noise flow). Experiments show an average 25dB SNR over 10 trials, each of which contains 250x10 samples with the gain of 24, sampled at 250Hz. In comparison, the commercial DAQ product, g.USBamp ($15k) shows 21.35dB SNR with hardware filter off and 24.61dB SNR with hardware filter on (1-30Hz).

To quantify the internal noise in AFE analog circuit, we analyze the input-referred noise (IRN) with short positive and negative analog inputs. Experiment shows 0.785μV Vpp (peak to peak) and 0.126μV Vrms (root mean square) input referred noise under the test configuration as reported in *ADS1299* manual, 1000 samples at gain of 24 sampled and 250Hz sampling frequency with a 0.01Hz-70Hz bandpass filter applied. This measure IRN of EEGu2 is slightly smaller than 0.98μV Vpp and 0.14μV Vrms reported in *ADS1299* manual. Fig. 12 depicts the trend of decreasing IRN with larger channel gain setting (equivalently reducing internal noise). This indicates the analog noise is injected after the amplification.

3.2 Firmware Quality

We assess quality in terms of real-time and end-to-end delay.

3.2.1 AFE Communication Real-time Analysis

Fig. 13 depicts the AFE communication protocol reading samples from AFE. When DRDY signals data being available on the AFE chip, the processor reads data via SPI. AFE periodically generates new data overwriting old data, which imposes a hard real-time constraint, i.e. a deadline of reading samples within a sampling period. Since the SPI transaction takes fixed time (at a clock rate), the response time t_{delay} from DRDY to SPI read, is critical. OS scheduling contributes to this latency in that the processor may prioritize other processes.

Fig. 14 illustrates the cumulative distribution (CDF) of DRDY to CS delay in both kernel driver and user driver (the initially developed driver in user space only). Each driver is evaluated with and without additional system load (4 dummy applications of busy loops). The sampling frequency is set to 250Hz (4ms deadline). In Fig. 14, The x-axis is the delay time and y-axis shows the cumulative percentage of samples with a smaller delay. The user driver performance deteriorates rapidly with system load. Without system load, the delay of 90% samples ranges from 20μs to 40μs. However, the maximal delay of 4488μs indicates sample lost. When system loaded, more than 50% of the samples fail 4ms deadline due to the driver competing with other user processes for OS schedul-

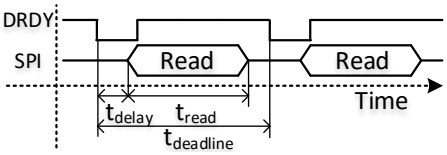


Figure 13: AFE Communication [9]

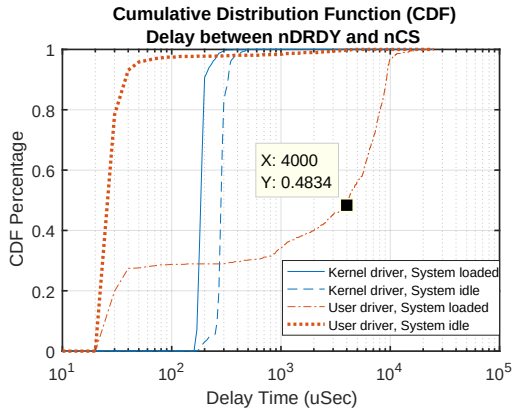


Figure 14: EEGu2 AFE Real-time Analysis

ing. Hence, this user driver is not useable.

In contrast, kernel driver shows an average delay of 189.40μs latency with system load and 287.22μs otherwise (see Table 3). The two tight curves indicate that the kernel driver is independent of the system load due to the higher priority of kernel process than user processes. Note that kernel driver even exhibits a better performance with system load because of the BBB frequency scaling (lower frequency when idle). Interestingly enough, the kernel driver average delay is much larger than the user driver. The additional delay of kernel driver results from the overhead of IIO subsystems, e.g. the two-level interrupt handling. Since the maximal delay of kernel driver is 520μs at full processor speed, the sampling rate can go up to 1KHz without failing the real-time requirement.

Latency (Unit: μs)	Mean	STD	Max
Kernel Driver w/o Load	287.22	37.84	626.00
Kernel Driver w/ Load	189.40	22.51	520.00
User Driver w/o Load	74.44	353.92	4488.00
User Driver w/ Load	4229.05	3805.45	20756.00

Table 3: EEGu2 AFE Real-time Analysis

3.2.2 *Firmware End-to-End Delay*

End-to-end delay measures the delay from the time an analog signal appears at the input to the time the sample can be read at the DAQ HW API. This response time of EEGu2 evaluates how fast the EEGu2 responds to the environment (input change). The experiment result shows 10.7ms and 12.3ms for end-to-end delay of user driver (no system load) and kernel driver respectively. The additional delay of kernel driver stems from the IIO overhead, including interrupt handling and data buffering. While the end-to-end delay indicates the warm-up latency when acquisition starts, the kernel driver can keep up with up to 1KHz sampling (1ms period) during the data streaming (see Section 3.2.1).

3.3 Demo: BCI Speller

3.3.1 *Application*

People with locked-in syndrome (LIS) can benefit from reliable assistive systems that help them regain their independence. Despite the loss of verbal communication and motor control, LIS individuals are fully conscious and aware of their surroundings. We developed a BCI speller using EEGu2 to help LIS individuals regain verbal communication. BCIs based on the steady state visually evoked potential (SSVEP) paradigm use the fact that focusing on a steadily flashing LED results in visual cortex EEG signals with the dominant frequency of the flickering and its harmonics [13].

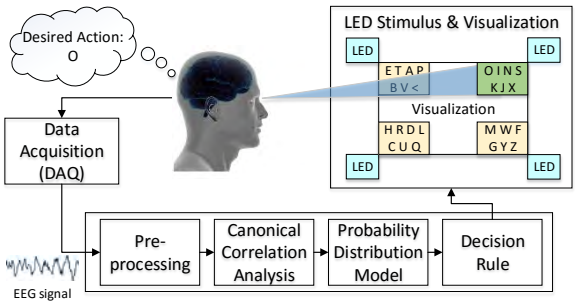


Figure 15: BCI speller processing flow

By pairing multiple LEDs (with different illumination patterns) to system actions (e.g. choosing a letter from the alphabet), the user's intent can be extracted by estimating the frequency of the attended stimuli from EEG.

Our application is executed in two stages: calibration/training and running. In the training stage, the software learns the EEG pattern when users focusing on each LED array. The physiological information is represented as canonical correlation scores between the EEG (20 4-second trials per stimuli) and linear combinations of sinusoids at the stimulus frequencies. This supervised learning is used to build a probability distribution of EEG evidence given the attended stimuli. In the running stage, the user is shown 4 icon boxes of letters associated with 4 LED arrays (8.1, 9.2, 10.3, and 11.4 Hz) respectively. The user can type by visually focusing on the flashing LED array that corresponds the desired letter (Fig. 15). The system will query the user in the minimum number of trials needed to achieve a confident decision (beyond a configuration threshold usually set to 0.85). The letter with the maximum posteriori inference will be chosen as the most probable one given all physiological (EEG) evidence. More details about the spelling application can be found in [14].

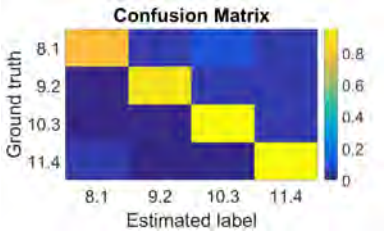


Figure 16: CCA Likelihood

The application algorithm is initially developed in MATLAB with the benefit of its algorithm development environment. EEGu2 records EEG and streams it to MATLAB for processing. Using the *HiLCPS Framework*, the application has hardware and location transparent access to EEGu2. Once the algorithm is validated in MATLAB, the application is automatically synthesized to C++ through domain-specific synthesis, which seamlessly interfaces with EEGu2 firmware. Since the interested EEG frequency is between 1 to 45Hz, EEGu2 is set to the lowest sampling frequency 250Hz for power saving and higher accuracy. Fig. 16 shows the confusion matrix as estimated via 10-fold cross validation. The confusion matrix shows how each frequency gets confused with others. The y-axis lists the ground truth and x-axis lists the estimates. The matrix shows the performance of the estimation. A diagonal matrix means the perfect performance. The results show an average of 90% accuracy in the intent inference process.

4. RELATED WORK

Commercial off-the-shelf DAQs are available on the market. The g.USBamp [15] is a high-end bio-signal acquisition system (cost about $15K) with 24 bits resolution and 16 channels. Companies like Emotive [16] and Neurosky [17] provide affordable low-end bio-signal acquisition devices mostly for gaming. Neurosky provides one channel EEG DAQ with 12-bit resolution at only $99. Emotive offers products such as EPOC+ with 14-channel 16-bit acquisition. With education license and raw data access permit, EPOC+ costs $3K. However, none of the above mentioned DAQs provides embedded processing accessible to users. In contrast, our EEGu2 is an affordable, battery powered bio-signal acquisition and processing system. EEGu2 provides 16-channel, 24-bit acquisition (equivalent to high-end g.USBamp) at only $600 for prototype and $400 for 1K unit production.

OpenBCI [7] offers an open source platform for BCI. The OpenBCI also uses *ADS1299* for acquisition. OpenBCI supports up to 16 channels by stacking a slave board over the master board (daisy chaining two *ADS1299* with identical configuration). OpenBCI contains an on-board microcontroller (50MHz PIC32MX250F128B) used only for data transmission, while a computer is still needed for data processing. In comparison to OpenBCI, our EEGu2 integrates two *ADS1299* that can support independent channel and sampling frequency configuration through two dedicated SPIs. Furthermore, EEGu2 offers much more powerful processing capability by using BBB (ARM Cortex-A8). The sufficient processing capability and the battery power supply enables a rapid embedded deployment of BBCI applications.

5. CONCLUSION

EEGu2, to our knowledge, is the first open embedded platform for both body/brain signal acquisition and embedded processing with mobile operations. As a modular design, EEGu2 comprises a customized cape on top of BBB, offering the 16-channel, 24-bit acquisition and powerful processing capability. The intelligent power management allows up to 12 hours battery operation. The results show the accuracy of EEGu2 acquisition (25dB SNR) competitive with state-of-the-art commercial DAQ (e.g. g.USBamp), while at a much lower cost ($400 vs $15K). In addition, the embedded processing of EEGu2 allows an embedded deployment of BBCI applications. The EEGu2 firmware is integrated into our *HiLCPS Framework* that enables the hardware and location transparent access via the MATLAB interface. This transparency simplifies the algorithm development in MATLAB, and most importantly, the prototyped BBCI application in MATLAB can be automatically deployed onto EEGu2 through domain-specific synthesis, dramatically increasing the productivity. We show the benefit of EEGu2 with a speller application. The results show an average of 90% accuracy in the intent inference process.

Future work aims to further improve the real-time response, as well as reducing cost and size.

Acknowledgment

This material is based upon work supported by the National Science Foundation under grant 1136027. The authors would like to thank Alex Zorn, David Crozier, David Karol and Matt Wood for developing first generation EEGu. We also thank James Gordon, Chris Horner, Chris Pittsley, Patrick Willett and David Winiarski for developing EEGu2 by improving EEGu.

6. REFERENCES

[1] J. Liu and Y. Zhou. Design of a Novel Portable ECG Monitor for Heart Health. In *2013 Sixth International Symposium on Computational Intelligence and Design (ISCID)*, volume 2, pages 257–260, October 2013.

[2] K. P. Thomas, A. P. Vinod, and C. Guan. Enhancement of attention and cognitive skills using EEG based neurofeedback game. In *2013 6th International IEEE/EMBS Conference on Neural Engineering (NER)*, pages 21–24, November 2013.

[3] S. Shenjie, K. P. Thomas, S. K. G, and A. P. Vinod. Two player EEG-based neurofeedback ball game for attention enhancement. In *2014 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, pages 3150–3155, October 2014.

[4] K. D. Katyal, M. S. Johannes, S. Kellis, T. Aflalo, C. Klaes, T. G. McGee, M. P. Para, Y. Shi, B. Lee, K. Pejsa, C. Liu, B. A. Wester, F. Tenore, J. D. Beaty, A. D. Ravitz, R. A. Andersen, and M. P. McLoughlin. A collaborative BCI approach to autonomous control of a prosthetic limb system. In *2014 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, pages 1479–1482, October 2014.

[5] F. Galán, M. Nuttin, E. Lew, P. W. Ferrez, G. Vanacker, J. Philips, and J. del R. Millán. A brain-actuated wheelchair: Asynchronous and non-invasive BrainŰcomputer interfaces for continuous control of robots. *Clinical Neurophysiology*, 119(9):2159–2169, September 2008.

[6] Umut Orhan. *RSVP KeyboardTM : An EEG Based BCI Typing System with Context Information Fusion*. Ph.D., Northeastern University, United States – Massachusetts, 2013.

[7] OpenBCI. OpenBCI: Dedicated to open-source innovation of human-computer interface technologies. http://www.openbci.com/, 2016.

[8] Gunar Schirner, Deniz Erdogmus, Kaushik Chowdhury, and Taskin Padir. The Future of Human-in-the-loop Cyber Physical Systems. *IEEE Computer*, 46(1):1–8, 2013.

[9] Texas Instruments Incorporated. ADS1299: Low-Noise, 8-Channel, 24-Bit Analog Front-End for Biopotential Measurements. http://www.ti.com/product/ADS1299, 2016.

[10] Shen Feng, Fernando Quivira, and Gunar Schirner. Framework for rapid development of embedded human-in-the-loop cyber-physical systems. In *IEEE 16th International Conference on BioInformatics and BioEngineering*, 2016.

[11] ISO - International Organization for Standardization. IEC 60601-1-11:2015. http://www.iso.org/, 2016.

[12] Analog Devices, Inc. Linux Industrial I/O Subsystem. https://wiki.analog.com/software/linux/docs/iio/iio, 2016.

[13] Danhua Zhu, Jordi Bieger, Gary Garcia Molina, and Ronald M. Aarts. A Survey of Stimulation Methods Used in SSVEP-based BCIs. *Intell. Neuroscience*, 2010:1:1–1:12, January 2010.

[14] M. Higger, F. Quivira, M. Akcakaya, M. Moghadamfalahi, h nezamfar, M. Cetin, and D. Erdogmus. Recursive Bayesian Coding for BCIs. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, PP(99):1–12, 2016.

[15] G.TEC MEDICAL ENGINEERING GMBH. g.USBamp - gtec highest accuracy biosignal data acquisition and processing system. http://www.gtec.at/, 2016.

[16] Emotiv, Inc. Emotiv EPOC / EPOC+. https://emotiv.com/, 2016.

[17] NeuroSky. EEG and ECG Biosensor Solutions. http://neurosky.com/, 2016.