# A NOVEL APPROACH FOR CONFIGURING THE STIMULATOR OF A BCI FRAMEWORK USING XML

**Indar Sugiarto**
Department of Electrical Engineering – Petra Christian University
Jl. Siwalankerto 121-131, phone: +62-31-2983442
Email: indi@petra.ac.id

***Abstrak***

*Dalam BCI (Brain-Computer Interface), setiap aspek harus diperhatikan demi keberhasilan operasional dari sistem BCI tersebut. Termasuk didalamnya adalah proses pembuatan stimulator BCI yang handal dan fleksibel, terutama stimulator yang berkaitan erat dengan umpan balik dalam bentuk aplikasi dari sistem BCI. Makalah ini menjelaskan pendekatan baru untuk membuat stimulator visual yang fleksibel dengan memanfaatkan format XML (Extensible Markup Language) yang dapat diterapkan pada sebuah unit sistem BCI. Dengan menggunakan format XML untuk mengatur konfigurasi dari stimulator visual sebuah unit BCI, kita dapat mengembangkan aplikasi BCI yang mampu mengakomodasi banyak strategi percobaan dalam penelitian tentang BCI. Unit BCI dan platform konfigurasinya dibuat dengan menggunakan bahasa pemrograman C++ dan memanfaatkan XML parser dari Qt yang bernama QXmlStream. Dari hasil implementasi dan pengujian terlihat bahwa file konfigurasi XML dapat dieksekusi dengan baik oleh sistem BCI yang digunakan. Selain kemampuannya dalam menghasilkan frekuensi kedipan yang fleksibel serta pengaturan format teks untuk sistem BCI berbasis SSVEP, file konfigurator tersebut juga memberikan pilihan pemakaian hingga 3 bentuk bangun, 16 warna, dan 5 indikator umpan balik yang berbeda. Metode yang dipaparkan dalam makalah ini dapat dipergunakan untuk meningkatkan kegunaan dari unit BCI yang telah ada saat ini seperti BF++ Toys dan BCI 2000.*

***Kata kunci**: Stimulator BCI, file konfigurasi, XML*

***Abstract***

*In a working BCI framework, all aspects must be considered as an integral part that contributes to the successful operation of a BCI system. This also includes the development of robust but flexible stimulator, especially the one that closely related to the feedback of a BCI system. This paper describes a novel approach in providing flexible visual stimulator using XML which has been applied for a BCI (brain-computer interface) framework. Using XML file format for configuring the visual stimulator of a BCI system, we can develop BCI applications which can accommodate many experiment strategies in BCI research. The BCI framework and its configuration platform is developed using C++ programming language which incorporate Qt's most powerful XML parser named QXmlStream. The implementation and experiment shows that the XML configuration file can be well executed within the proposed BCI framework. Beside its capability in presenting flexible flickering frequencies and text formatting for SSVEP-based BCI, the configuration platform also provides 3 shapes, 16 colors, and 5 distinct feedback bars. It is not necessary to increase the number of shapes nor colors since those parameters are less important for the BCI stimulator. The proposed method can then be extended to enhance the usability of currently existed BCI framework such as BF++ Toys and BCI 2000.*

***Keywords**: BCI Stimulator, Configuration File, XML*

## 1. INTRODUCTION

One of the most promising applications of biomedical electronics in the area of human-machine interaction is the so-called BCI (brain-computer interface). The BCI system comprises of many components including hardwares, softwares and protocols. The protocol in the BCI will be used to drive the system in order to ensure that the brain elicite necessary signals to be

acquired by the system. Depending on the type of the BCI, the present of a stimulator as a mean of the protocol implementation may be mandatory. The stimulator plays vital role in BCI system, especially for BCIs which operate exogenously. The so-called VEP (visual evoked potential) is a special potential which can be elicited using visual stimulator. Some researches show that certain pattern on this stimulator may produce different results, thus providing extra flexibilities for the stimulator will enrich the BCI research. It is also preferable that the number of trials in every BCI system must be set as small as possible and it means that the stimulator, especially the one which provides the visual stimuli, must operate in the very exact time and in the predetermined fashion.

These conditions, which are related to the manipulation of visual stimulator parameters, need to be well defined and adjusted to fit the general requirements in the BCI program. Unfortunately, most of the research groups involved with BCI "do" it in different ways according to not only the systems and platforms used but also to data file formats. Moreover, these are relative not only to the electrophysiological processed data (e.g. EEG, ERP signals) but also to configuration settings (e.g. feedback rule, stimulator behavior, etc). This fact represents an obstacle in tools exchange among different laboratories which perform similar BCI research. Also, it is practically impossible to simulate the behavior of a system and to optimize it by assembling modules from different groups as a unique way for describing their characteristics.

A typical situation is a group which is involved in the implementation of spellers that would like to simulate the performances of systems built by assembling their developed modules with other ones available from the literature. A recent study [2] has demonstrated that under certain conditions it is possible to reliably predict the behavior of a system built by assembling different modules if a well-defined and relatively simple description of them is available. This will allow the optimization of many BCI systems without the need of really building them: a way to break the interdependence of the modules and thus of the various research groups. An effort of providing such "bridge" in the BCI research area has been started by [5] which describe file formats based on the XML technology for storing some entities frequently encountered among the BCI community as well as some free tools (BF++ Toys) that use them and that were developed to optimize the performances of complete systems and to document them. Unfortunately, they only focused their proposed method for only storing the experiment data. Indeed, in order to be fully accessible to a wide audience, however, it is also necessary to provide an efficient and extensible file format platform for the stimulator part of a BCI program in order to fulfill the needs of virtually any scientist and to provide tools that are able to handle the situation. Also, as pointed out by Wenke Burde [3], most BCI research tends to improve BCI performance through developing better signal processing algorithms but only a few which consider the aforementioned bridging mechanism between independent BCI researchers as challenging opportunity. Maybe it is quite difficult for BCI researchers to find appropriate methods or protocols which provide the necessary platform for building a working BCI framework.

This paper deals with this issue, i.e. providing a platform for configuring the visual stimulator of a BCI system in the form of XML (eXtensible Markup Language) file format. In our proposed platform, we store the stimulator configuration data (the numbers of stimuli, frequencies, etc.) in XML files in order to allow fast and simple interchanging of them, even on Web, and at the same time a robust and customizable way of formatting them. This paper is written in the following systematic. After a brief introduction about the background situation, a review about the BCI framework will be described and the XML short introduction will be provided prior to the explanation about the proposed platform. Afterward, issues on software implementation and testing will be explained. Finally, the paper will be closed with conclusions.

## 2. METHOD

### 2.1. XML and the Development of the BCI Framework

Extensible Markup Language (XML) is a general-purpose specification for creating personal markup languages, which makes use of tags. It describes data and distributes them in a format, which is independent from the platform. This independence comes from the fact that XML does not use a specific language: in fact, XML tags are not predefined, so that everyone can write his own personal tags. XML is very similar to HTML (Hyper Text Markup Language) but it is not its substitute as its aim is different: XML was designed for storing and exchanging

data, while HTML was created for showing data in a format easily readable even if hardly adjustable during time. In this way, it is very comfortable sharing data across different laboratories as every researcher is free to extend them with the results of an analysis without disturbing the activity of the other ones that can receive the new data and continue to work with the previously developed tools. The essential characteristic of XML is that data are independent. The content of a file is kept separate from its presentation, so that one can store the content in an XML file only once and then extract and visualize it in the desired format (according to the final format, there is an appropriate XML technology that allows its generation, called XSLT).

BCI system tries to create a direct bridge between human CNS (central nervous system) with a computer or machine through neurophysiologic signals generated by the brain. It then creates a new pathway for the brain to carry its message. At some extent, it is better to describe a BCI system as a hybrid system in which all of its components collaborate together to form an integrated system, as written in IEEE Signal Processing Magazine Volume 25 Number 1, January 2008: "A brain computer interface is a system that includes a means for measuring neural signals from the brain, a method/algorithm for decoding these signals and a methodology for mapping this decoding to a behavior or action". This description of BCI system sounds more technical since it emphasizes the importance of measurement-decoding-mapping interaction within the system. The following diagram shows how those three components work to construct a BCI system [6]. As Fig 1 show, the measurement task is performed within the Source sub-system, decoding is performed by Signal Processing sub-system and mapping tasks are performed by the User Application. Operator module acts as a central relay for system configuration and online presentation of results to the investigator.

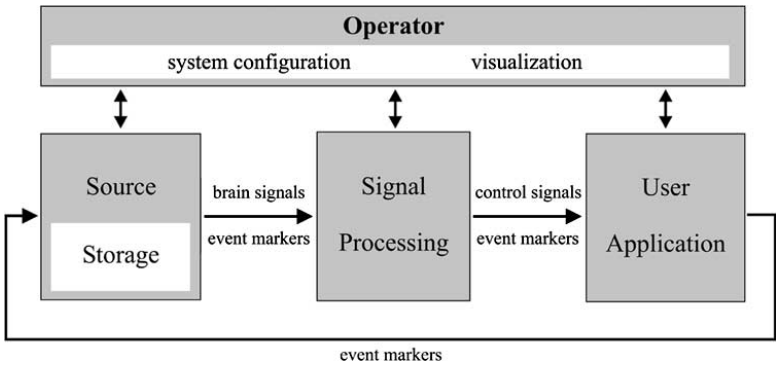



Fig 1. Mechanism of a BCI system in general. During operation, information (i.e., signals, parameters, or event markers) is communicated from source to signal processing, to user application, and back to source.

Those three core components of any BCI system require specific amount of computer processing resource in order to work as ideal as possible as in "dream" BCI. In order to maintain its complexity, it is necessary to develop a framework which encapsulates all those three core components and provides standard protocols for their communication. An important remark about this framework is that, as we already know, the human nervous system (including the brain) is the most complex and adaptive system in the world. This title is reflected by the phenomenon that every human brain may react differently to the same BCI system. There are subjects who have good response on certain type of BCI system but the other ones don't have it. For some subject oscillations in the alpha and beta band during motor imagery work better than the evoked potential approach and vice versa. This inter-subject variability corresponds to spatial patterns and spectrotemporal characteristics of brain signals. A good BCI framework should also address this issue. An example of excellent approach for this matter is given by BCI2000 program [6], which can be viewed as a BCI framework as well. The BCI2000 system tries to make a flexible BCI framework by collaboration of many user defined sub-programs. Unfortunately, this system requires every sub-program to be hard-coded and embedded into the main program using low-level inter-process communication (IPC) parameters before one can use the BCI2000 in the real application. This "not so user-friendly" approach will become an

obstacle in bridging independent systems of each BCI laboratories. Thus in this paper, in order to accommodate the future online-based BCI system, we develop the BCI framework using the same approach as [5].

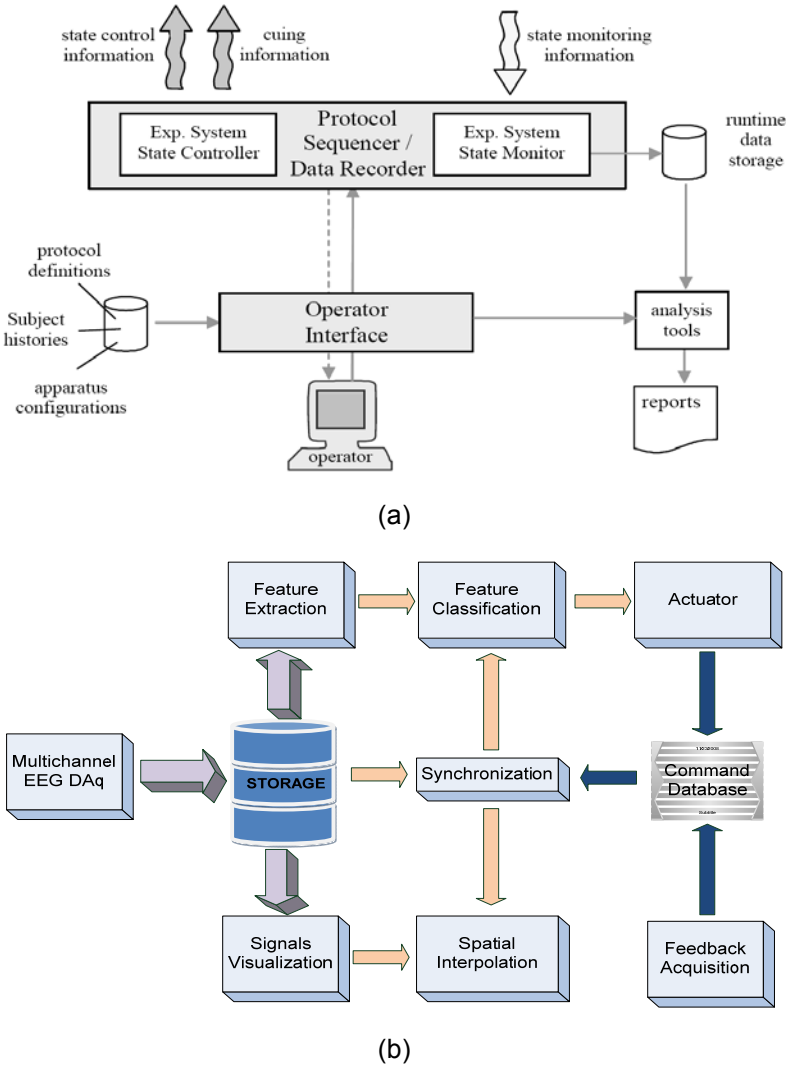



Fig 2. Block diagram of the BCI framework. (a) in system setup scenario, (b) for data abstraction

Since the development of BCI framework requires excessive part of program codes, the explanation in this paper will be focused on the visual stimulator part of the BCI framework. Based on the human capability to respond to external stimuli, we can expect that BCI could give feedback in many ways. Sometimes it is quite difficult to determine which type of feedback is best for certain application of a BCI system. Even presenting more than one type of stimulus would help subject to improve his/her ability to control his/her own EEG. For example, combining both random pattern and flickering pattern will provide a multimodal approach for the BCI system enhancement. Fig 2 shows our proposed BCI framework in system setup scenario and the data abstraction level. As it can be seen in Fig 2, there are at least three components of sub-systems in the BCI framework system setup scenario which work simultaneously: data recorder, signals visualization (analysis tools), and operator interface. But in the data abstraction layer, there are at least seven major components which are required to be synchronized: data acquisition, signals database/storage, feature processing (extraction and classification), visualization (including stimulator), command generation for actuator, command database, and the feedback acquisition.

This research modifies the [7] and [5] methods in order to comply with our proposed method and the entire system works as follows. Since it is possible to predict a collection of CIs (control interfaces), given a collection of TRs (transducers), the performances of all the system that is possible to build by combining all the TRs with all the CIs can be clearly used to optimize BCI systems because it will be sufficient to choose the most efficient combination of TRs and CIs. All the entities used for the description of TRs and CIs will be written in XML format both for storage necessity and for easily managing the simulations for the computation of the performances indicators.

The TR is composed by the acquisition stage, which deals with the electrophysiological signals, and by the Classifier, formed by the Feature Extraction and the Feature Classification, whose task is to extract the features of interest from the signals and to translate them into a logical symbol (LS) which belongs to the classifier logical alphabet (LA). The LS, in general, has no semantic meaning but is just a mapping with some subject's performance (e.g. the character selection in the spelling program [8]), the selection of a row in P300 virtual keyboard task [4]). The CI encodes sequences of LSs and turns them into semantic symbols (SS) that belong to a semantic alphabet (SA) by the Actuator part that can be used to drive an external peripheral (e.g. virtual keyboards, robots, neuroprostheses). As an example, if we have a logical alphabet of four symbols α, β, γ, δ the CI can implement an encoding for a virtual keyboard application that associates the sequence ααα to the semantic symbol A, then ααβ to the semantic symbol B and so on (usually one of the LSs is reserved for the UNDO key).

In this way 27 semantic symbols can be encoded with 3 LSs-long sequences so that the 26 characters of the English alphabet plus the space one can be mapped. In addition to these symbolic parameters, the so-called Extended Confusion Matrix (ECM) metric for the evaluation and optimization of the performances of BCI systems which combines the information about the entities previously described (LA, SA, Encoders) with those derived from a new component, is used. We used XML for storing LAs, SAs, ECMs and Encoders in the following format:

```
<?xml version="1.0" encoding="UTF-16"?>
<DOC>
        <Type Name="Subject_01" SubType="Session_01"/>
                <ECM>
                        <Row>
                                <In>α</In>
                                <C1>393</C1>
                                <C2>4</C2>
                                <C3>4</C3>
                                <C4>4</C4>
                                <Abstentions>45</Abstentions>
                        </Row>
                        <Row>
                                <In>β</In>
                                <C1>8</C1>
                                <C2>381</C2>
                                <C3>8</C3>
                                <C4>8</C4>
                                <Abstentions>45</Abstentions>
                        </Row>
                        <Row>
                                <In>γ</In>
                                <C1>16</C1>
                                <C2>16</C2>
                                <C3>357</C3>
                                <C4>16</C4>
                                <Abstentions>45</Abstentions>
                        </Row>
                        <Row>
                                <In>δ</In>
                                <C1>32</C1>
                                <C2>32</C2>
                                <C3>32</C3>
                                <C4>309</C4>
                                <Abstentions>45</Abstentions>
                        </Row>
                </ECM>
        </DOC>
```

In the visualization part, the EEG signals can be displayed spontaneously (in time domain) or being interpolated for spatial visualization. After the features are extracted, certain translation algorithm is required to decode the feature into executable commands for actuator. In order to record the activity of the actuator, a command database program is required to keep track and store the commands. After a certain mental task has been given, the subject brain will react in a certain manner which will produce specific rhythmic signals. These feedback signals are then being recorded and/or processed for further analysis. In order to synchronize all works in the framework, we develop a synchronization protocol which generate timing signals and request/acknowledge signals for data interchange between sub-systems.

### 2.2. The Proposed Platform for Stimulator Configuration

When working with exogenous BCI system, an external stimulator is required to elicit certain EEG pattern. In a visual-based BCI stimulator, there are general requirements for displaying the cues. For example in SSVEP-based BCI, the general parameters will be the flickering animation frequencies, sizes, colors, shapes, positions, feedback style, and patterns (solid or textured). To accommodate these, we develop the XML file format as follows.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ssvep_p300>
<ssvep_p300 version="2.0">
        <define shape=user_defined_shape>
                <self user_defined_parameters></self>
                <bar user_defined_parameters></bar>
                <label user_defined_parameters></label>
                <texture user_defined_parameters></texture>
        </define>
</ssvep_p300>
```

As the label name implies (i.e. ssvep_p300), we develop the stimulator so that it can work for SSVEP- and P300-based BCI or combination of both BCI types.

Description:

- The XML file must have document ID as ssvep_p300 and it must contain attribute version=”2.0”. The first line, which consists of statement <?xml version=”1.0” encoding=”UTF-8”?> may be omitted but it is not recommended.
- Every ‘define’ element will start a new flickering animation definition and its shape is must be specified in the ‘shape’ attribute. The following possible values for ‘shape’ attribute are: rectangle, triangle, and circle. Note that the attribute’s value is case insensitive.
- After defining flickering animation’s shape, the next step is creating four elements within the ‘define’ element. These four elements are ‘self’, ‘bar’, ‘label’, and ‘texture’.
- ‘self’ element will be used to describe the flickering animation itself and it contains several attributes. These attributes, which reside in an element definition, may be specified in random order. The following attributes are valid for ‘self’ element:
    - size → will determine the physical size of the animation in pixels;
    - posx and posy → will determine the top-left corner for square and circle flickering animation and determine the first point for triangle animation;
    - posx2, posy2, posx3, posy3 → will be used exclusively for determining the second and the third points of a triangle flickering animation;
    - freq → will determine the frequency of flickering animation;
    - color → will determine flickering animation’s color; possible values are: white, red, green, blue, cyan, magenta, yellow, gray, darkred, darkgreen, darkblue, darkcyan, darkmagenta, darkyellow, darkgray, and lightgray
    - texture → will determine if the flickering animation has texture or not;
- ‘bar’ element will be used to specify the appearance and behavior of feedback bar for each flickering animation. The following attributes are valid for ‘bar’ element:
    - type → will determine the style of feedback bar; possible values are: none, individual, halfleft, halfright, and combination

- color → will determine the color of feedback bar; possible values are: white, red, green, blue, cyan, magenta, yellow, gray, darkred, darkgreen, darkblue, darkcyan, darkmagenta, darkyellow, darkgray, and lightgray
- brush → will determine the filling rule for the bar; possible values are solid and line
- size → will determine the width of the bar; the height of the bar is follows the size of the flickering animation;
- frame → will determine if the feedback bar is displayed with a frame or not, so the possible values are yes or no
- framecolor → if the feedback bar has a frame, then this attribute will determine its color; the possible values are the same with another color definition; if this attribute is ignored, then the frame color will be set the same color with its parent
- blinking → will determine whether the feedback bar should also blinking with the same frequency as the flickering animation itself; however, it is recommended to set this value as no to avoid performance degradation when using many animations
- threshold → will determine the threshold value of feedback bar;

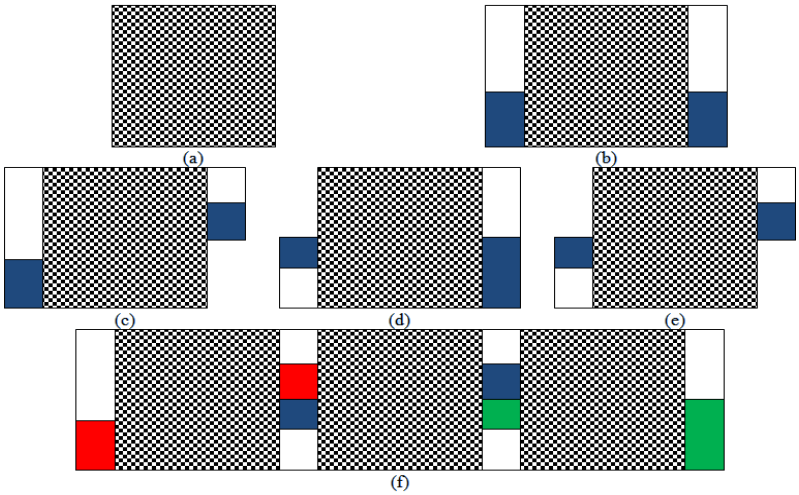
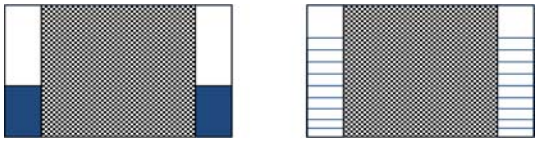



Fig 3. Several possible feedbacks displays: (a) NO_FEEDBACK_BAR, (b) INDIVIDUAL, (c) HALF_RIGHT, (d) HALF_LEFT, (e) COMBINATION, and (f) a result for combining several feedback types.



(a) (b)

Fig 4. Two possible appearances of feedback bar: solid (left) or discrete (right).

– 'label' element will be used to specify flickering animation's label and it will be displayed anywhere on the screen, depending on the specified values below:
  - text → any ASCII text in this attribute is valid as a flickering animation's label; if this attribute is NULL, then the flickering animation will be displayed without label
  - size → will determine the font size of the label;
  - color → will specify the color of the text; the possible values are the same with the other color's attribute above
  - posx and posy → will determine the exact position of the text; if these attributes are negative, then the text will be displayed on the center of the flickering animation;

- 'texture' element will be used to specify flickering animation's texture. If textures are going to be used for flickering animation animations, the texture attribute in the 'self' element must be set as 'on' first. The following attributes are valid for this element:
  - first → specify the full-path location of the first texture image filename; any recognizable image by Qt may be used
  - second → specify the full-path location of the second texture image filename; any recognizable image by Qt may be used
- Every element must be closed with appropriate closing tag. Otherwise, it will not be recognized as an element in the XML document format. This XML document must be saved in standard ASCII format (with extension .xml).

In our proposed method, we also develop several type of feedback bars for SSVEP-based BCI. Using the configuration file explained above, the following feedback bars can be inserted together with the flickering stimulator for SSVEP-based BCI (can be seen in Fig 3 and Fig 4).

## 3. IMPLEMENTATION AND DISCUSSION

The program is developed using C++ in Qt's open source framework. The current version of Qt being used in this paper is Qt 4.3.3. Since the free-license Qt is used, Microsoft Visual Studio cannot be used as an IDE (Integrated Development Environment) for developing the program, instead, Eclipse version 3.3.1.1 (Europa version) with Qt-Integration plug-in is used. As an open-source mandatory of Qt and Eclipse, g++ version 3.4.5 of MinGW is used as the compiler. When using g++ as compiler, gdb can be used as the debugger.

Qt supports two models of XML implementation: SAX (Simple API for XML) and DOM (Document Object Model). The main different of SAX and DOM is SAX reports "parsing events" directly to the application through virtual functions, while DOM converts an XML document into a tree structure, which the application can then navigate. Qt also provides its own way to read or write XML which is called QXml module (which consists of several useful class such as QXmlAttributes, QXmlStreamReader, QXmlStreamWriter, etc). In this paper, QXmlStreamReader is used for reading configuration file.

After compiling the BCI framework program, we provide the XML configuration file for configuring the stimulator. For example, the following XML script will define a BCI application which consists of four flickering animations: two square-shape (e.g. the first one with texture and feedback bar and the second one with neither texture nor feedback bar), one textured circle-shape with feedback bar, and one textured triangle shape with feedback bar.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ssvep_p300>
<ssvep_p300 version="2.0">
	<define shape="rectangle">
		<self size="100" posx="0" posy="0" posx2="" posy2="" posx3="" posy3=""
			freq="13.5" color="white" texture="no"></self>
		<bar type="none" color="white" brush="solid" size="20" frame="no"
			framecolor="ignore" blinking="no" threshold=""></bar>
		<label text="" size="" color="" posx="" posy=""></label>
		<texture first="" second=""></texture>
	</define>
	<define shape="rectangle">
		<self size="100" posx="200" posy="0" posx2="" posy2="" posx3=""
			posy3="" freq="15.0" color="yellow" texture="yes"></self>
		<bar type="halfleft" color="white" brush="line" size="20" frame="yes"
			framecolor="red" blinking="yes" threshold="0"></bar>
		<label text="two" size="20" color="green" posx="200" posy="50"></label>
		<texture first="default_texture1.bmp" second="default_texture2.bmp">
		</texture>
	</define>
	<define shape="triangle">
		<self size="100" posx="0" posy="200" posx2="100" posy2="200"
			posx3="50" posy3="300" freq="17.5" color="blue" texture="yes">
		</self>
		<bar type="individual" color="red" brush="solid" size="20" frame="yes"
			framecolor="red" blinking="no" threshold="200"></bar>
		<label text="three" size="20" color="green" posx="0" posy="250"></label>
```

```
        <texture first="default_texture1.bmp" second="default_texture2.bmp">
        </texture>
</define>
<define shape="circle">
        <self size="100" posx="200" posy="200" posx2="" posy2="" posx3=""
              posy3="" freq="10" color="red" texture="yes"></self>
        <bar type="halfright" color="red" brush="solid" size="20" frame="yes"
              framecolor="red" blinking="no" threshold="200"></bar>
        <label text="this is a circle LED" size="20" color="yellow" posx="100"
              posy="200"></label>
        <texture first="default_texture1.bmp" second="default_texture2.bmp">
        </texture>
</define>
</ssvep_p300>
```

Fig 5 shows the result for the above configuration script.

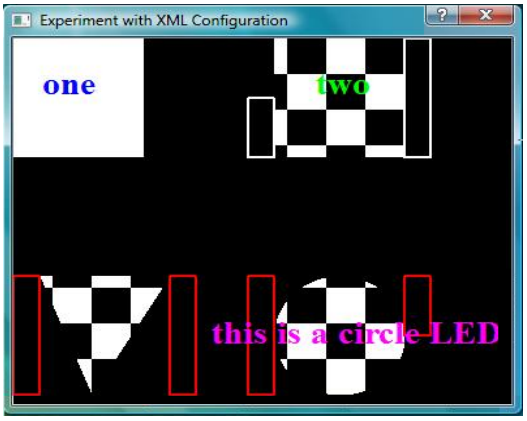



Fig 5. Display result from experiment with the given XML configuration script.

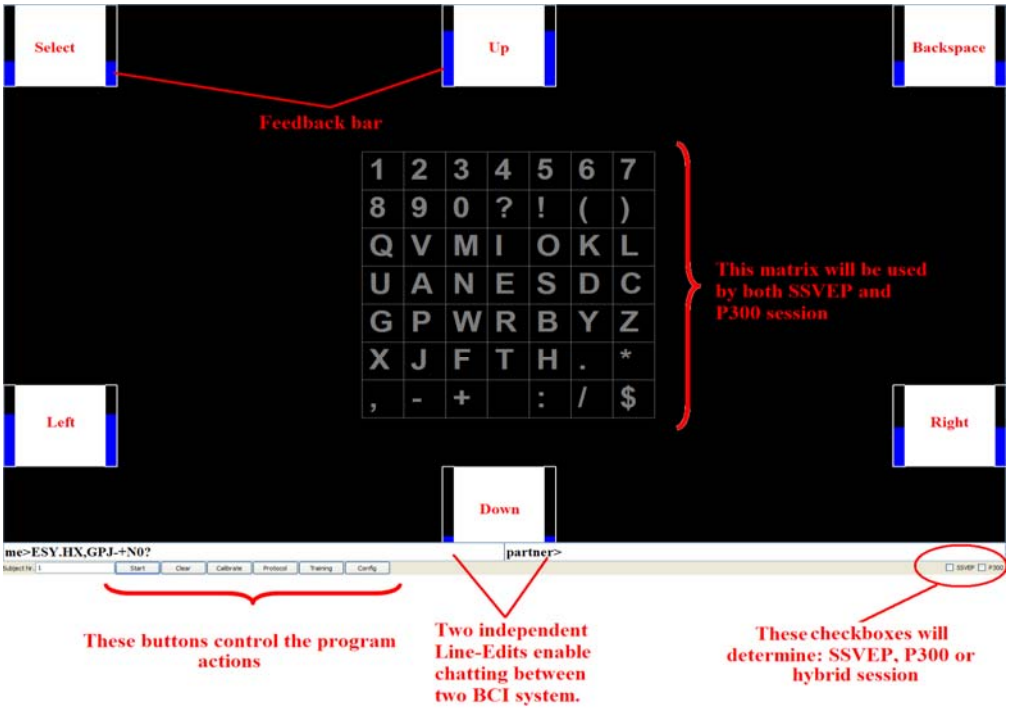



Fig 6. The complete BCI framework in action. The BCI framework uses XML file configuration to control all widgets on the screen.

The proposed method has been pre-evaluated and implemented in real application of BCI system for spelling program application. It has been demonstrated successfully as reported by [8] and [1] as a result of public experiment in the event of CeBIT 2008 in Hannover. Fig 6 shows the screenshot result of the proposed BCI framework and its configuration platform.

## 4. CONCLUSION

A novel approach in providing flexible visual stimulator using XML has been presented and applied for development of a BCI (brain-computer interface) framework. Using XML file format for configuring the visual stimulator of a BCI system, we can develop BCI applications which can accommodate many experiment strategies in BCI research. The BCI framework and its configuration platform is developed using C++ programming language which incorporate Qt's most powerful XML parser named QXmlStream. The implementation and experiment shows that the XML configuration file can be well executed within the proposed BCI framework. Beside its capability in presenting flexible flickering frequencies and text formatting for SSVEP-based BCI, the configuration platform only limited to providing 3 shapes, 16 colors, and 5 distinct feedback bars. It is not necessary to increase the number of shapes nor colors since those parameters are less important for the BCI stimulator. The proposed method can then be extended to enhance the usability of currently existed BCI framework such as BF++ Toys and BCI 2000.

## REFERENCES

[1] Allison, BZ et al., **"BCI Demographics I: How many (and what kinds of) people can use an SSVEP BCI?"**, Proceeding of 4th International Brain-Computer Interface Workshop and Training Course 2008, Verlag der Technischen Universität Graz – Austria, 2008.

[2] Bianchi L, Quitadamo LR, Garreffa G, Cardarilli GC, Marciani MG., **"Performances evaluation and optimization of Brain-Computer Interface systems in a copy spelling task"**, IEEE Transactions on Neural Systems and Rehabilitation Engineering, 15(2): 207-216, 2007.

[3] Burde W, and Blankertz B., **"Is the locus of control of reinforcement a predictor of brain-computer interface performance?"**, Proceedings of the 3rd International Brain-Computer Interface Workshop and Training Course 2006, Verlag der Technischen Universität Graz: 76-77, 2006.

[4] Millán, J., **"Adaptive Brain Interfaces for Communication and Control"**, 10th International Conference on Human-Computer Interaction, Greece, 2003.

[5] Quitadamo LR, Marciani MG, Bianchi L., **"Optimization of Brain Computer Interface systems by means of XML and BF++ Toys"**, International Journal of Bioelectromagnetism, 9(3): 172- 184, 2007.

[6] Schalk G, McFarland DJ, Hinterberger T, Birbaumer N, Wolpaw JR., **"BCI2000: a general-purpose brain–computer interface (BCI) system"**, IEEE Transaction on Biomedical Engineering, 51(6):1034–43, 2004.

[7] Sugiarto I, and Handoyo I., **"Application of Distributed System in Neuroscience, A Case Study of BCI Framework"**, Proceeding of The First International Seminar on Science and Technology (ISSTEC 2009), Universitas Islam Indonesia, Jogjakarta, 2009.

[8] Sugiarto I, Allison BZ, Gräser A., **"Optimization Strategy for SSVEP-Based BCI in Spelling Program Application"**, Proceeding of International Conference on Computer Engineering and Technology 2009 (ICCET 2009), International Association of Computer Science and Information Technology (IACSIT), Singapore, 2009.