Proceedings of the Second International Conference on Computing Methodologies and Communication (ICCMC 2018)
IEEE Conference Record # 42656; IEEE Xplore ISBN:978-1-5386-3452-3

Designing a Brain Computer Interface Using EMOTIV Headset and Programming Languages

Ruida Zeng[1 3], Ajay Bandi[2 3], Abdelaziz Fellah[2 3]
[1]Missouri Academy of Science, Mathematics and Computing
[2]School of Computer Science and Information Systems
[3]Northwest Missouri State University
Maryville, MO, United States 64468
dozeng@vip.qq.com[1], ajay@nwmissouri.edu[2], afellah@nwmissouri.edu[3]

Abstract*—In this paper, we explore the bio sensory headset EMOTIV EPOC+ in the context of brain computer interfaces which record the brain signals and convert them into keystrokes. Reinforced by programming language constructs, these signals were able to trigger the movements of the finch robot and switch lightbulb objects. The observation and analysis of our case studies using human subjects using these brain computer interfaces shows two problems, subject's frustration and the time-consuming of learning to train the device. More experiments should be conducted in the future on different types of devices to explore the match between the brain signals and the actual actions. Also, investigate the experience of the users in learning to design and develop the brain computer interfaces.

***Keywords*— Brain Computer Interface, programming, human-computer interaction, user interfaces, Emotiv**

I. Introduction

Brain-Computer Interface (BCI), a science fiction vision some years ago, have become almost a reality and infiltrated many aspects of our life. There is no uniform definition of what is called Brain-Computer Interfaces (BCIs), but we will use the definition as defined in [1]: "A BCI is a direct communication pathway between a human brain and computer systems." That is, a technology that records the brain activity, acquires signals that reflects the user's intention, and interacts with a computer or machine by sending commands to carry out some actions. Recording and measuring the brain signals rely on different mediums and techniques such as the non-invasive widely used electroencephalography (EEG) for recording the brain signals from specific regions of the head by attaching electrodes along the scalp [4, 5]. BCIs have multiple applications in diverse fields such as neuro science, education, rehabilitation, entertainment, video games, and research. There are still many challenges in developing reliable and robust BCIs which are not only rooted in brain science, but linked to several other areas such as machine learning, BCI devices, and technologies.

In this research, we explore one particular BCI device, the EMOTIV EPOC+ headset, to develop a brain-computer interface to enable a user to interact back and forth with a moving robot that turns ON and OFF the LED lights. In addition, one major goal of this study is to integrate computer-programming languages (i.e., Java, C++, Snap!) into the BCI technology and bring this challenge to the classroom. In addition, we develop and reinforce the EMOTIV brain computer interface by using Java, C++ and Snap! programming languages to control the Finch robot to act upon a set of selective commands given by the user.

II. Related Work

Conventional input devices such as the keyboard and the mouse have dominated human-computer interface (interaction) (HCI) and are used as an interface to communicate between a user and a computer. However, these devices rely on voluntary muscle movements which may not be present in users with a partial or complete loss of physical autonomy. Other technologies such as pen-based systems, haptic devices, and speech recognition software, which enable interactions through pen pressure, finger touches, and voice recognition, have been used in HCI [8, 9, 10]. With current advances in neuro science and BCI technologies, a large number of BCI systems and a vast set of applications have been being investigated and developed in research. BCI technologies focused on several aspects such controlling input/output devices, communicating channels, acquiring signal techniques, and processing signals. Several recent prototypes already enable users to navigate in virtual environments, construct manipulate virtual objects, or play games just by means of their cerebral activity [3]. Some other recent studies have also investigated EEG signals in problem solving and creative design process [4, 5, 6].

The rest of the paper is organized as follows. Section II gives an overview of the EMOTIV EPOC+ headset hardware for recording and processing signals, controlling Finch robot and turn on and off LED lightbulb using BCI headset. Then, we present the results of the phase by training the device to accomplish the designated task, controlling the Finch robot.
Section III discusses the results, section IV discusses the conclusion and future work.

III. Case Studies

We conducted two case studies to explore the usage of BCI in the programming courses. The two case studies are controlling the Finch robot and turn ON and OFF of a LED bulb using a BCI headset through brain signals. In our case studies, we chose EPOC+ headset manufactured by EMOTIV. The rationale of using this device is unlike other BCI devices, the EPOC+ headset, is portable and uses wireless connections to the computer. It is rechargeable using Internal Lithium Polymer battery (640mAh) that can last from 6 to 12 hours continuously, compatible with Windows, OSX, Linux, Android, and iOS platforms. It has 14 non-invasive electrodes reads electroencephalograms (EEG) data off the user's brain [2]. These are important factors because we are looking for a robust BCI device that can adapt to most user's unique situations.

The EPOC+ headset is commonly used for contextualized researchers and other advanced BCI applications. In this research, we will not focus on the exploration of the neuroscience aspect of this device nor to develop a highly advanced BCI device-based application. There will be two rudimentary applications using programming languages. The first uses the EPOC+ headset to control a Finch robots, and the second uses the headset to control a LED lightbulb.

A. Preliminary Setup

The setup of the device includes the headset, USB PC transceiver dongle (which needed to be inserted into a computer in order for the user to receive wireless transmission from the headset), saline hydration fluid, 16 electrode assemblies, a hydrator case, a pair of rubber ear pad with electrode body, and USB cable [2]. Following the quick start guide provided in the package, we charged the headset completely for 4 hours before hydrating the sensors using saline fluid. Then, we installed the sensors into each of the black plastic arms of the headset. After that, we were able to fit the headset by sliding the headset down from the top of the head using both hands. We used the EMOTIV Xavier Control Panel 3.3, a companion software, to monitor the headset [2]. It guides the fitting and connecting of the headset in a more detailed manner, shows the electrodes contact quality, sets base lines (eyes open/eyes closed), and provides access to a few detection algorithms.

The EPOC+ headset is illustrated in the Figure 1. This device has 14 electrode sensor channels that are used to detect brain signals at the time the user wears the device on the head. The electrodes are named as AF3, F7, F3, FC5, T7, P7, O1, O2, P8, T8, FC6, F4, F8, and AF4. These electrodes need be hydrated using saline solution for a better detectability and readability of the brain signals.

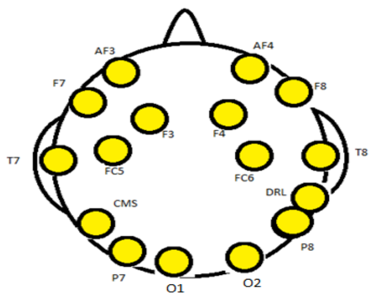


Figure 1: Positions of the Electrodes

The headset includes a Common Mode Sense (CMS) active electrode at the P3 location an absolute voltage reference. It also includes a Driven Right Leg (DRL) passive electrode at the P4 location as a feedback cancellation system to float the reference level on the common mode body potential [2]. The signals from CMS and DRL are neither recorded nor estimated. They are only used as a relative reference point as part of the standard referencing protocol [2]. The positions of these electrodes are demonstrated by a user in Figure 2. A green dot indicates a strong connection between the brain and the device, meaning that the device is effectively reading the brain signals, an orange/red dot indicates weaker connections, while a black dot indicates an unstable connection. The user must get all the electrodes to show a strong connection before proceeding to the next step.

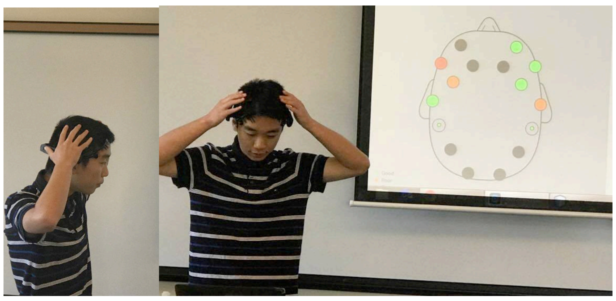
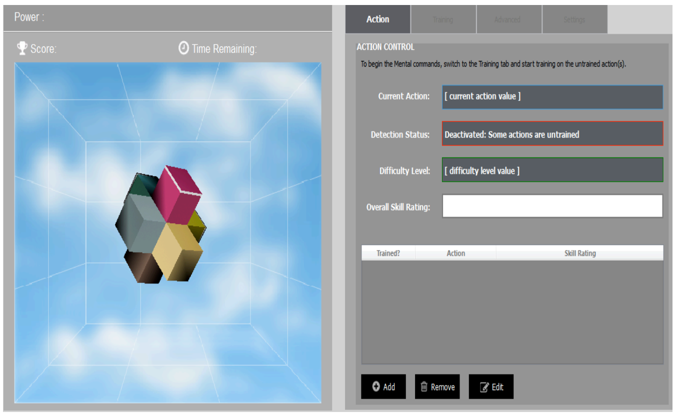


Figure 2: A Subject Wearing the EPOC+ Headset

There are three different kinds of accurate machine learning algorithms offered by EMOTIV through the Xavier Control Panel: performance metrics, facial expressions, and mental commands [2]. The mental commands feature, the one used in our case studies, is an interface for the users to train the system to recognize thought patterns related to various desired outcomes [2]. For calibrating and training, we were asked by the machine to perform certain actions to the virtual block through direct mental commands as illustrated in the Figure 3.



ACTION CONTROL
To begin the Mental commands, switch to the Training tab and start training on the untrained action(s).

Current Action: [current action value]
Detection Status: Deactivated: Some actions are untrained
Difficulty Level: [difficulty level value]
Overall Skill Rating:

Trained?	Action	Skill Rating

Figure 3: EMOTIV Training Interface

As for the machine-human interaction aspect, the EPOC+ is one of the easiest BCI device to operate. Typically, when people encounter a device, they face two distinct gulfs, the Gulf of Execution, where the users figure out how to use it, and the Gulf of Evaluation, where the users figure out what state it is in and whether their actions are able to get them to their goal [7]. The seven stages of action are further specified as shown in Figure 4. There are five fundamental psychological concepts when we interact with the BCI headset, the most important one being affordance, which is the relationship between the physical properties of an object and a person, determines what actions are possible [7]. The EPOC+ headset's quick set up and easy-to-understand instruction manual made the affordance much easier to discover, which in terms of solidifying the bridges so the users can proceed with the stages of action quicker and more concisely.

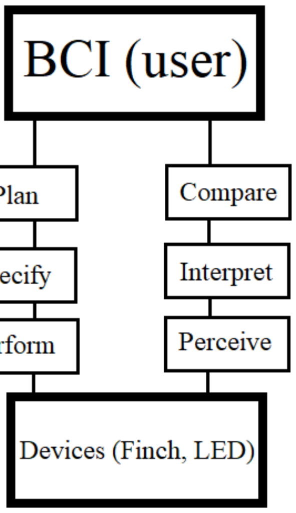


Figure 4: Seven Stages of Action Cycle

The next subsections explain and demonstrate two other possible affordances of the EPOC+ headset using, two illustrations, the EPOC+ headset to move robots and blink the LED light bulb.

B. Controlling the Finch robot using the BCI headset

We used the Finch robot, a small mobile robot designed by BirdBrain Technologies for students learning computer science by providing them a physical representation of the source code.

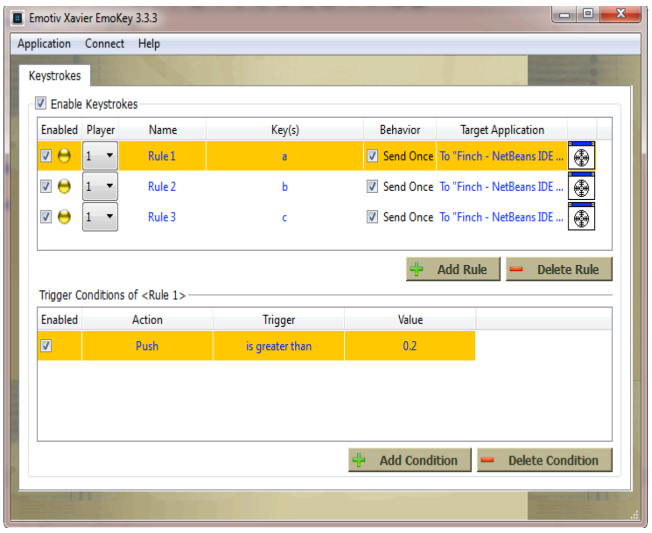
In order to train the device to accomplish the designated task, to control the Finch robot, another custom software EmoKey is used. EmoKey is a software provided in the EMOTIV Xavier Control Panel that allows users to link their mental commands to keystrokes and mouse operations within the host machine's existing applications. We use the programming languages Java and Snap! to accomplish this task.

First, we programmed the Finch robot to act upon a selective command given by the user using Java. An object of type Scanner was used to read characters from the consoles and used a series of control structures to make the Finch move. Each input letter, either, "a", "b" and "c" with each will lead to the Finch Robot to either move forward for 10 seconds, move backward for 10 seconds, or not to move at all, and it will also shine two different colors of light depending on input given, as shown in the source code (Figure 5).

```
package Code.emotivFinch; // Needs a package
declaration to move to another folder

import
edu.cmu.ri.createlab.terk.robot.finch.Finch;
import java.util.*;

/** * Created by: Ruida Zeng * Date:
11/16/2017 * A file to control the Finch using
EMOTIV headset */

public class EmotivFinchBasic {

public static void main(final String[] args) {

// Instantiating the Finch object
        Finch emotivFinch = new Finch();
//Set LED to blue and ask for input command
        emotivFinch.setLED(255, 255, 0);
Scanner finchScanner = new Scanner(System.in);
        emotivFinch.saySomething("Please give
one command");
        System.out.print("Command: ");
        String userInput =
finchScanner.next();
        do {
            if (userInput.equals("a")) {
                emotivFinch.setLED(0, 255, 0);
//Set Finch LED to green and move it forward

emotivFinch.setWheelVelocities(255,255,2000);
                emotivFinch.saySomething("The
Finch is moving forward as commanded");
            } else if (userInput.equals("b")){
                emotivFinch.setLED(0, 255, 0);
//Set Finch LED to green and move it backward

emotivFinch.setWheelVelocities(-255,-255,
2000);
                emotivFinch.saySomething("The
Finch is moving backward as commanded");
            } else {
                emotivFinch.setLED(0, 255, 0);
//Set Finch LED to red
                emotivFinch.saySomething("The
Finch is not moving as commanded");
                emotivFinch.sleep(2000);  //No
movement
            }
            emotivFinch.saySomething("Please
give one command");
            System.out.print("Command: ");
            userInput = finchScanner.next();
        } while (userInput.equals("c") ||
userInput.equals("b") ||
userInput.equals("a"));
// Always end your program with finch.quit()
        emotivFinch.quit();
        System.exit(0);
    }
}
```

Figure 5: Java source code in Java NetBeans IDE 8.2 for controlling the Finch robot

To establish an association between signals from the EPOC+ headset and the Java compiler that controls the Finch robot, we wrote three rules that send keystrokes to target applications. We corresponded the triggered conditions, such as "push", "pull" and "lift" (which we have trained earlier as shown in Figure 4) to the keystrokes "a", "b" and "c", respectively. For example, rule 1 of the setup in EmoKey is shown below in Figure 6. Each keystroke is sent only once based on each signals. In addition, we changed the target application in EmoKey to the desired application, which is the Java compiler in this case, in order to avoid confusion or sending the keystrokes to the wrong application. Rule 1 from Figure 6 illustrates that a strong push signal (greater than the default value of 0.2) from the headset send keystroke "a" to the Java compiler, triggering the Finch robot to move forward. Rule 2, which is not shown in Figure 6 (trigger conditions), moved the Finch backwards if it detects a strong pull signal from the headset by sending keystroke "b" to the Java compiler. Rule 3, which is also not shown in Figure 6 (trigger conditions), terminates the program without the Finch robot moving if it detects a strong lift signal from the headset by sending keystroke "c" to the Java compiler.



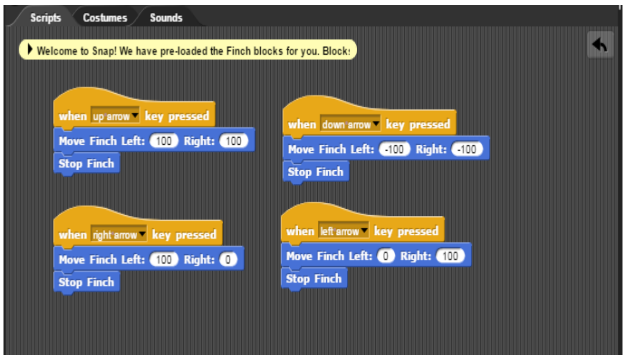
Enabled	Player	Name	Key(s)	Behavior	Target Application
✓	1	Rule 1	a	✓ Send Once	To "Finch - NetBeans IDE ...
✓	1	Rule 2	b	✓ Send Once	To "Finch - NetBeans IDE ...
✓	1	Rule 3	c	✓ Send Once	To "Finch - NetBeans IDE ...

Trigger Conditions of <Rule 1>

Enabled	Action	Trigger	Value
✓	Push	is greater than	0.2

Figure 6: EmoKey for Java Finch

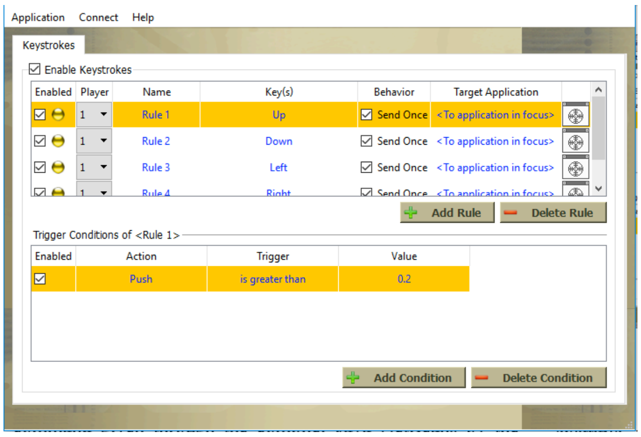
Our next step is to program the Finch Robot to move according to the user's real-time keyboard input. For example, when the user presses the up arrow, the robot should move forward, it should also stop moving forward when the user releases the up arrow. Same events should happen for the other three directions, corresponding to the down, left, and right arrows respectively. We programmed source codes in Snap! that can do the mentioned tasks, as shown in Figure 7.


Scripts
Costumes
Sounds
Welcome to Snap! We have pre-loaded the Finch blocks for you. Block:

```
when up arrow key pressed
Move Finch Left: 100 Right: 100
Stop Finch

when down arrow key pressed
Move Finch Left: -100 Right: -100
Stop Finch

when right arrow key pressed
Move Finch Left: 100 Right: 0
Stop Finch

when left arrow key pressed
Move Finch Left: 0 Right: 100
Stop Finch
```

Figure 7: Snap! source code for controlling the Finch robot

Then, we set up four rules for this application and the corresponding trigger conditions including, "push", "pull", "lift", and "drop" by using the upward, downward, left, and right arrow keys, respectively. Rule 1 of the EmoKey set up is also illustrated in Figure 8. Once more, each keystroke is only sent once based on each signals. Then, we set up the conversions in EmoKey shown in the Figure 8. The Finch Robot runs based on the brain signals tht EmoKey receives and convert to various different keys. Rule 1 in Figure 8 illustrates that a strong push signal from the headset indicates a pressing of the up arrow key to the online Snap! triggering the Finch robot to move forward. When the strong push signal disappears, the up arrow key is released. Rule 2, which is not shown in Figure 8 (trigger conditions), moves the Finch backwards if it detects a strong pull signal from the headset by sending the down arrow key to the online Snap! interface. When the strong pull signal disappears, the down arrow key is released. Rule 3, which is also not shown in Figure 8 (trigger conditions), turns the Finch robot to the left direction if it detects a strong lift signal from the headset by sending the left arrow key to the online Snap! interface. When the strong lift signal disappears, the left arrow key is released. Rule 4, which is also not shown in Figure 8 (trigger conditions), turns the Finch robot to the right direction if it detects a strong drop signal from the headset by sending the left arrow key to the online Snap! interface. When the strong drop signal disappears, the right arrow key is released.



Enabled	Player	Name	Key(s)	Behavior	Target Application
✓	1	Rule 1	Up	✓ Send Once	<To application in focus>
✓	1	Rule 2	Down	✓ Send Once	<To application in focus>
✓	1	Rule 3	Left	✓ Send Once	<To application in focus>
✓	1	Rule 4	Right	✓ Send Once	<To application in focus>

Trigger Conditions of <Rule 1>

Enabled	Action	Trigger	Value
✓	Push	is greater than	0.2

Figure 8: EmoKey for Snap! Finch

C. Displaying LED light using the BCI headset

Our next goal is essentially to use the EPOC+ headset and integrate it into users' daily life. Internet of Things has a prominent role in using devices to collect the data while transmitting and receiving from one device to another, such as using the device to control the lighting of the room, the movement of wheelchair, or other electronic activities remotely.

In this case study, we activated a blinking LED lightbulb using EPOC+ headset-controlled LED light with the help of EmoKey.

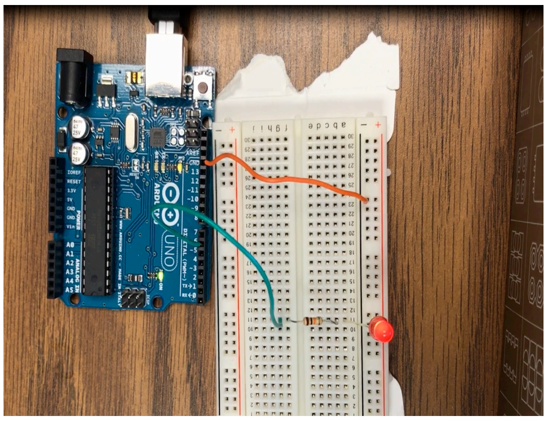
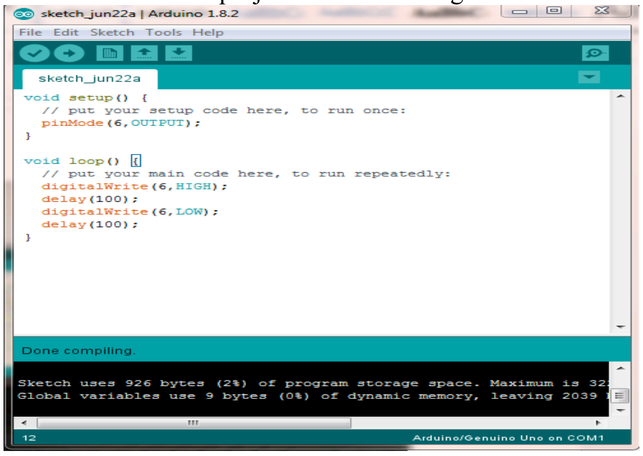


Figure 9: Circuits for LED light bulb

We set up an electric circuit using an Arduino Uno and a breadboard, connected to the laptop via USB port, shown by Figure 9. We used Arduino IDE and the programming language C++ to blink the LED rapidly in a loop with intervals of 400 milliseconds (on, wait, off, wait). The short code written for this project is shown in Figure 10.


sketch_jun22a | Arduino 1.8.2
File Edit Sketch Tools Help
sketch_jun22a

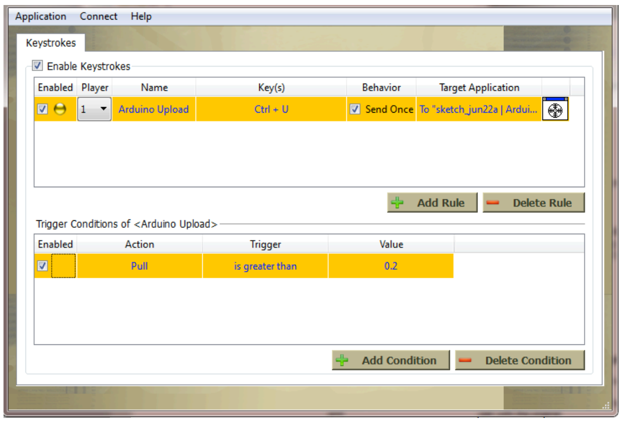
```
void setup() {
  // put your setup code here, to run once:
  pinMode(6,OUTPUT);
}

void loop() {
  // put your main code here, to run repeatedly:
  digitalWrite(6,HIGH);
  delay(100);
  digitalWrite(6,LOW);
  delay(100);
}
```

```
Done compiling.

Sketch uses 926 bytes (2%) of program storage space. Maximum is 32
Global variables use 9 bytes (0%) of dynamic memory, leaving 2039
```

12
Arduino/Genuino Uno on COM1

Figure 10: C++ source code in Arduino IDE for controlling the blinking LED light bulb

As shown in Figure 11, the only rule set up in EmoKey is to correspond the trigger condition of "pull" to the hotkey "Ctrl + U", which means "upload" or to start running the written program in Arduino IDE. Thereby, by using the brain signal "push", we were able to command the LED to light up and to blink rapidly.



☑ Enable Keystrokes

Enabled	Player	Name	Key(s)	Behavior	Target Application
☑	1	Arduino Upload	Ctrl + U	☑ Send Once	To "sketch_jun22a | Ardui...

Trigger Conditions of <Arduino Upload>

Enabled	Action	Trigger	Value
☑	Pull	is greater than	0.2

Figure 11: EmoKey for the C++ LED

RESULTS AND DISCUSSIONS

Our goals are achieved in a manner that we successfully associate the EPOC+ headset with simple codes in Snap!, Java, and C++, we were able to train our brain signals to both move a Finch robot and to switch lightbulbs on and off. The operation of this could be complex device has been simplified to a level that is appropriate for the community with computer programming knowledge.

We briefly demonstrated the EPOC+ headset to a group of twenty students majoring in computer science. Most students showed interest in this device with some showing particularly high interest. These human-computer interaction experiences showed that this device is an innovative and exciting idea. However, from our case study setup on test subjects on the usage of this interfaces during demonstrations caused frustration of learning to train the EPOC+ headset. However, the students are excited to work with the headset.

CONCLUSIONS AND FUTURE WORK

Our contributions in this paper are two-fold. 1) Controlling Finch robot using the BCI device, 2) turning on and off the LED lightbulb using the BCI device. Using the crucial EmoKey feature of the EMOTIV device, we are able to devise a multitude of ways in a fashion that only requires basic programming knowledge as demonstrated by exemplary simple applications in our case studies.

For future studies on BCI headsets, we plan to investigate BCI mental commands to control a larger scope of objects in the real world by combining other devices and embedded into the third party applications. Since EMOTIV also provides a subscription based software that EPOC+ provides access to dense array, high quality, raw EEG data—EMOTIV Xavier Pure EEG. Using Pure EEG, we were able to collect five sets meaningful data recorded by the EPOC+ headset using Pure EEG, namely EEG, PPT, Gyro, Motion, and Data Packet, which represented the range of possible data that the EMOTIV headset can collect. Future studies will further use these knowledge and data sets to develop more objective-specific tasks. Using these SDKs provided by EMOTIV, we can take the raw data recorded and use it directly to perform outcome actions, which will strengthen the match between brain signals and actions performed.

ACKNOWLEDGMENT

We would like to thank Dr. Michael Rogers for his help in setting up the circuits.

REFERENCES

[1] Abdulkater, S. N., Ayman, A., Mostafa, M. S. M., Brain-computer interfacing: applications and challenges, *Egyptian Informatics Journal*, *16*(2), 213-230, 2015.

[2] EPOC, Product description, Retrieved from https://www.emotiv.com/epoc/?gclid=EAIaIQobChMImeyV4L7m1wIVQp7ACh2UawWxEAAYASAAEgLhWPD_BwE

[3] Esfahani, E. T., Sundararajan, V., Classification of primitive shapes using brain-computer interfaces, *Computer-Aided Design*, *44* (10), 1011-1019, 2012.

[4] Pun, T., Alecu, T.I., Chanel, G., Kronegg, J. & Voloshynovskiy, S., Brain-computer interaction research at the computer vision and multimedia laboratory, University of Geneva. *IEEE Transactions on Rehabilitation Engineering*. 210-213.

[5] Li M, Lu BL. Emotion classification based on gamma-band EEG. In: *Proceedings of Annual International Conference of the IEEE Engineering in Medicine and biology society EMBC* 2009. 1123-6, 2009.

[6] Gavin J. Dollman , Lizette De Wet , & Tanya R. Beelders, Effectiveness with EEG BCIs: Exposure to traditional input methods as a factor of performance, *Proceedings of the South African Institute for Computer Scientists and Information Technologists Conference*, October 07-09, 2013, East London, South Africa

[7] Norman, D., The Design of Everyday Things, New York, NY: Basic Books, 2013.

[8] Vallabhaneni, A., Wang, T. and He, B., Brain—computer interface. *In Neural Engineering* Springer, Boston, MA, 85-121, 2005.

[9] Bandi, A. and Heeler, P. Usability testing: A software engineering perspective, *Proceeedings of the 2013 International Conference on Human Computer Interactions (ICHCI)* , Aug 23, 1-8, 2013.

[10] Brandman, D. M., Hosman, T., Saab, J., Burkhart, M. C., Shanahan, B. E., Ciancibello, J. G., & Kelemen, J., Rapid calibration of an intracortical brain-computer interface for people with tetraplegia. *Journal of Neural Engineering,* 15(2), 2018.