VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“Jnana Sangama”, Belgaum 590014, KARNATAKA, INDIA



Project Report

On

 “BCOT (Brain Computer Of Things)”

Submitted in Partially fulfilment of the requirement for the award of degree

Of

Bachelor of Engineering

In

Computer Science & Engineering

Of Visvesvaraya Technological University, Belgaum.

Submitted by:

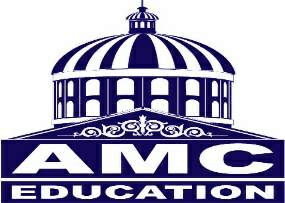
**SHREYAS S K (1AM16CS163)**

**SHUBHAM KUMAR (1AM16CS165)**

Under the guidance of:

**Mrs. SONALI S DASH**

(Assistant Prof., Dept. of CSE*)*

******

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

AMC ENGINEERING COLLEGE

(NAAC & NBA Accredited, Approved by AICTE, New Delhi & Affiliated to VTU, Belagavi)

 18th K.M. Bannerghatta Main Road, Bengaluru – 560083

Year 2019-20

Accredited by NAAC & NBA, MINISTRY OF HRD, NEW DELHI & Affiliated to VTU Belagavi)

https://lh4.googleusercontent.com/JH0wO0MhMi8MBR4E8avFlRqqGDdjKsP05L4Leh-zxdULIlYg6oJ3cD2y6bNHU5dOwdZlgft3so0lWM_5GnppsnQKxGnfZu78jJLm6h95kHy6qu_oPOtCbp81fvb7ezLPId5_we8F

**AMC ENGINEERING COLLEGE**

(NAAC & NBA Accredited, Approved by AICTE, New Delhi & Affiliated to VTU, Belagavi)

## Department of Computer Science & Engineering

CERTIFICATE

Certified that the Project work entitled **“** **BCOT ( Brain Computer Of Things) ”** carried out bya bonafide student, **SHREYAS S K (1AM16CS163) and SHUBHAM KUMAR (1AM16CS165)** of **AMC Engineering College**, in partial fulfillment for the award of **Bachelor of Engineering in Computer Science & Engineering of Visvesvaraya Technological University, Belgaum** during the year **2019-2020**. It is certified that all corrections / suggestions indicated have been incorporated in the report. The project report has been approved as it satisfies the academic requirements in respect of project work prescribed for the said Bachelor of Engineering degree.

|  |  |  |
| --- | --- | --- |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| Signature of Project Guide | Signature of HOD | Signature of Principal |
| **Mrs. Sonali S Dash**  **Asst. Prof., Dept. of CSE,**  **AMCEC** | **Dr. Latha C. A.**  **Prof., & HOD, Dept. of CSE,**  **AMCEC** | **Dr. A. G. Nataraj**  **Principal,**  **AMCEC** |

**External Viva**

**Name of the examiners Signature with date**

**1. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**2. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

DECLARATION

We, Shreyas S K and Shubham Kumar, the undersigned student of 8th semester Department of Computer Science and Engineering, AMC Engineering College, declare that our project work entitled “BCOT (Brain Computer Of Things)” is a bonafide work of ours. Our project is neither a copy nor by means a modification of any other engineering project.

We also declare that this project was not entitled for submission to any other university in the past and shall remain the only submission made and will not be submitted by us to any other university in the future.

|  |  |  |
| --- | --- | --- |
| Name | USN | Signature |
| SHREYAS S K  SHUBHAM KUMAR | 1AM16CS163  1AM16CS165 | **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of this project report would be complete only with mention of people who made it possible, whose support rewarded my effort with success.

First and foremost, We would like to thank God, the Almighty for being so merciful on me. He guided me in each and every walk of life to do something good and hence this dissertation.

We consider it our privilege to express our deep gratitude to our Chairman Dr. K. R. Paramahamsa for providing the infrastructure and an opportunity to peruse our degree in his prestigious institution.

We owe our gratitude to our Principal, Dr. A. G. Nataraj, for his kind motivation and support during the project.

We express our heartfelt gratitude and humble thanks to Dr. Latha C. A., Professor & HOD of Department of Computer Science & Engineering, for the facilities and support during the project.

We greatly express our gratitude to our guide Mrs. Sonali S Dash, Assistant Professor, Department of Computer Science & Engineering, for her valuable suggestions and guidance regarding the project.

We greatly express our gratitude to our project co-ordinator, Dr. Doddegowda B. J., Associate Professor, Department of Computer Science & Engineering, for taking a keen interest in the progress of the project and suggestions given to me.

We also would like to acknowledge all the teaching and non-teaching staff of the department for their kind cooperation during the project.

In the end, We sincerely thank our beloved family members and friends for their valuable suggestions, encouragement and unwavering support.

SHREYAS S K (1AM16CS163)

SHUBHAM KUMAR (1AM16CS165)

ABSTRACT

BCOT (Brain Computer Of Things) is a collaboration between a brain and machine learning that enables signals from the brain to direct some external activity, such as control of a computer or a mobile or any other devices just by the thought. The interface enables a direct communications pathway between the brain and the object to be controlled. In the case of cursor control, for example, the signal is transmitted directly from the brain to the mechanism directing the cursor, rather than taking the normal route through the body's neuromuscular system from the brain to the finger on a mouse.

Our project is a derivative of BCI technology , one of the biggest challenges in developing BCI technology has been the development of electrode devices and/or surgical methods that are minimally invasive. In the traditional BCI model, the brain accepts an implanted mechanical device and controls the device as a natural part of its representation of the body . Most important thing is our project is creative derivative of BCI technology through our project anyone can do anything just by a thought.

Keywords:

* Brain Computer Interface (BCI)
* Electroencephalogram (EEG)
* Cortex API
* Brainwaves
* PyCharm

CONTENTS

|  |  |
| --- | --- |
| **DECLARATION** | **i** |
| **ACKNOWLEDGEMENT** | **ii** |
| **ABSTRACT** | **iii** |
| **1. INTRODUCTION** | **1** |
| **1.1 Aim and Objective** | **3** |
| **1.1.1 Aim** | **3** |
| **1.1.2 Objective** | **3** |
| **1.2 Problem Statement** | **3** |
| **1.3 Existing System** | **4** |
| **1.4 Proposed System** | **5** |
| **1.4.1 Advantages** | **5** |
| **2. LITERATURE SURVEY** | **6** |
| **2.1 An EEG-based BCI System for 2-D Cursor Control** | **7** |
| **2.2 Using Brain Computer Interface for Home Automation** | **7** |
| **2.3 BCI Virtual Keyboard For Accessibility** | **8** |
| **3. REQUIREMENTS SPECIFICATION** | **9** |
| **3.1 Hardware Requirements** | **10** |
| **3.2 Software Requirements** | **10** |
| **4. SYSTEM DESIGN AND ARCHITECTURE** | **11** |
| **4.1 Design** | **12** |
| **4.1.1 System Architecture** | **12** |
| **4.2 Flowchart** | **15** |
| **4.3 Modules** | **17** |
| **4.3.1 Cortex API** | **17** |
| **5. IMPLEMENTATION** | **21** |
| **5.1 Languages and Platform Used For Implementation** | **22** |
| **5.1.1 Python** | **22** |
| **5.2 Software Used For Implementation** | **24** |
| **5.2.1 PyCharm** | **24** |
| **5.3 Code** | **27** |
| **5.3.1 Implementation code** | **27** |
| **5.4 Observation and Result** | **62** |
| **6. CONCLUSION AND FUTURE WORK** | **64** |
| **6.1 Conclusion** | **65** |
| **6.2 Future Work** | **65** |
| **REFERENCES** | **67** |
| **APPENDIX** | **68** |
| **Appendix –A Snapshots** | **69** |

LIST OF FIGURES

|  |  |  |
| --- | --- | --- |
| **Figure 1** | **Block Diagram of System Architecture** | **12** |
| **Figure 2** | **Setup Tab** | **14** |
| **Figure 3** | **Technical Specifications** | **15** |
| **Figure 4** | **Flowchart of API Flow** | **16** |
| **Figure 5** | **Facial Expressions** | **63** |
| **Figure A.1** | **Emotiv insight device** | **69** |
| **Figure A.2** | **Home page** | **69** |
| **Figure A.3** | **Output of Blink for ON/OFF** | **70** |
| **Figure A.4** | **Features page** | **70** |
| **Figure A.5** | **Output for Facial Expressions** | **71** |
| **Figure A.6** | **Output for Act To Speak** | **71** |

**INTRODUCTION**

**CHAPTER 1**

**INTRODUCTION**

The human brain — intelligent and unique — is challenging scientists, who are determined to decode its complexity and unlock possibilities to enhance human lives. By harnessing artificial intelligence (AI), they have already made breakthroughs in manmachine interactions through Watson, Siri, and more. But, for AI to have a truly transformational impact, artificial neural networks need to be further reinforced by human native intelligence. While the human brain finds ways to exceed our physical capabilities, the combination of mathematics, algorithms, computational methods, and statistical models is accelerating our scientific pursuit. Artificial Intelligence (AI) gathered momentum after Alan Mathison Turing developed a mathematical model for biological morphogenesis, and authored a seminal paper on computing intelligence. Today, AI has grown from data models for problem-solving to artificial neural networks — a computational model based on the structure and functions of human biological neural networks.

BCOT (Brain Computer Of Things) is a collaboration between a brain and machine learning that enables signals from the brain to direct some external activity, such as control of a computer or a mobile or any other devices just by the thought. The interface enables a direct communications pathway between the brain and the object to be controlled. In the case of cursor control, for example, the signal is transmitted directly from the brain to the mechanism directing the cursor, rather than taking the normal route through the body's neuromuscular system from the brain to the finger on a mouse.

Current brain-interface devices require deliberate conscious thought; some future applications, such as prosthetic control, are likely to work effortlessly.

Our project is a derivative of BCI technology , one of the biggest challenges in developing BCI technology has been the development of electrode devices and/or surgical methods that are minimally invasive. In the traditional BCI model, the brain accepts an implanted mechanical device and controls the device as a natural part of its representation of the body . Most important thing is our project is creative derivative of BCI technology through our project anyone can do anything just by a thought.

**1.1 Aim and Objective**

**1.1.1 Aim**

The aim of this project is to create BCOT that enables signals from the brain to direct some external activity, such as control of a computer or a mobile or any other devices just by the thought. The interface enables a direct communications pathway between the brain and the object to be controlled. In the case of cursor control, for example, the signal is transmitted directly from the brain to the mechanism directing the cursor, rather than taking the normal route through the body's neuromuscular system from the brain to the finger on a mouse. The main aim of this project is to create an interface between brain and the computer that will allow an individual with severe motor disabilities to have effective control over external devices and perform some work independently.

**1.1.2 Objective**

Design a method for real time collection of different types of data from the headset. That could help in facial expression which will detect the face emotions like happy, sad, excitement etc. Our project is a new way to control cursor and lights in bcot application. This also helps in Act to speech:-this features has an artificial voice-speaks when muscular movements is made or thought about something, this could really help patients.

**1.2 Problem Statement**

The ultimate purpose of a direct brain control of things (BCOT) is to allow an individual with severe motor disabilities to have effective control over devices. A BCOT system detects the presence of specific patterns in a person’s ongoing brain activity that relates to the person’s intention to initiate control. BCOT offer a method to evaluate awareness and to restore a communication channel. People with ALS, spinal cord injury or brain-stem stroke represent populations for whom BCOT is important and useful. Hence there arises a need for faster, more reliable BCOT systems that work for more potential users. These serve as a problem statement for the project.

**1.3 Existing System**

“Brain machine interface (BMI) allows users to control external devices by analysing their brain signals”. BCIs are now used for experiments not just online but also data analysis offline by using open source toolboxes. This can offer people with severe disabilities a chance for ground breaking communication with the outside world. BMI is still at an early stage with only a small number of brain signal algorithms available, but this is an incredible opportunity for AI and human interaction. “Using collaborative BCI with the target of improving human response in Visual Target Recognition is currently in development”.

“One can achieve higher levels of perceptual and cognitive performance by leveraging the power of multiple brains through BCI collaboration”. Collaborative BCIs have the objective of improving human performance by using neural data from multiple brains in combination with advanced signal analytics meaning the more brains that are used the greater the performance with support vector machine (SVM) technology being utilised to classify the data. AI with quantum technology can for example be used for DNA mapping with the possibility of the human genes of strength and intelligence identified meaning BCIs can be used to implement and enhance these traits for future human development.

**1.4 Proposed System**

EEG devices used to be prohibitively expensive and time consuming to connect, and the data required expert knowledge to interpret. They required conductive gel to connect with very clean hair and skin. We hence chose emotive insight headset, which is the best option for a dry EEG sensor available in the market. The headset is easy to wear and works on dry skin without any gels. It only needs electrical contact with head. A BCOT system detects the presence of specific patterns in a person’s ongoing brain activity that relates to the person’s intention to initiate control .The main aim of this project is to create an interface between brain and the computer that will allow an individual with severe motor disabilities to have

effective control over external devices and perform some work independently.

The objectives of our project are:

• Design a method for real time collection of different types of data from the headset. That could help in facial expression which will detect the face emotions like happy, sad, excitement etc.

• New way to control cursor and lights in bcot application.

• Act to speech:-this features has an artificial voice-speaks when muscular movements is made or thought about something, this could really help patients.

The project attempts to leverage these properties of brain, cortex API and PyCharm to achieve an efficient BCOT system.

**1.4.1 Advantages**

1. New way to control cursor and lights in bcot application

2. This will help in creating a direct communication pathway between a human brain and any external devices like computers.

3. One of the biggest reasons why BCI is deemed as advanced technology is because it can make previously passive devices into ‘smart’ and active ones.

4. Cursor control, Act to speech etc.

**LITERATURE SURVEY**

**CHAPTER 2**

**LITERATURE SURVEY**

**2.1 An EEG-based BCI System for 2-D Cursor Control by Combining Mu/Beta Rhythm and P300 Potential**

**Yuanqing Li, Jinyi Long, Tianyou Yu, Zhuliang Yu, Chuanchu Wang, Haihong Zhang, Cuntai Guan**.

Two-dimensional cursor control is an important and challenging issue in electroencephalogram (EEG)- based brain computer interfaces (BCIs). To address this issue, here we propose a new approach by combining two brain signals including Mu/Beta Rhythm during motor imagery and P300 potential. In particular, a motor imagery detection mechanism and a P300 potential detection mechanism are devised and integrated such that the user is able to use the two signals to control, respectively, simultaneously and independently, the horizontal and the vertical movements of the cursor in a specially designed graphic user interface. A real-time BCI system based on this approach is implemented and evaluated through an online experiment involving six subjects performing 2-D control tasks. The results attest to the efficacy of obtaining two independent control signals by the proposed approach. Furthermore, the results show that the system has the merit compared with prior systems: it allows cursor movement between arbitrary positions.

**2.2 Using Brain Computer Interface for Home Automation**

**Harish Verlekar , Hrishikesh Gupta, Kashyap Joshi**

Brain computer interface (BCI) is a communication pathway between the brain and the external peripheral devices like computers. We propose to use this technology for Home automation. Home automation can be totally revolutionized using BCI. Brain produces various types of waves like alpha (9-13Hz), beta (14-30Hz), theta (4- 8Hz), delta (1-3Hz).Using these waves we can control various home appliances. The entire concept consists of 4 main stages detection, amplification, processing, output. First detecting the brain signals using an EEG cap or electrodes. These brain signals are very weak hence in second stage we need to amplify these brain signals to a usable amount and filter these to remove noise. Then thirdly, we will have to convert these signals into digital by using A to D converter and into a type a computer software or a microcontroller can understand. Fourth, taking this decoded signal and sending these signals wirelessly, by using an RF circuit to a distant switch circuit, which will turn on or off the appliance in vicinity. Using this technology the life of people would be further simplified, physical efforts would be considerably reduced and it would also prove as a boon for physically disabled people.

**2.3 BRAIN-COMPUTER INTERFACE VIRTUAL KEYBOARD FOR ACCESSIBILITY**

**Jonathan Corley, Dustin Heaton, Jeff Gray, Jeffrey C. Carver, Randy Smith**

This paper describes our experiences in building a virtual keyboard implemented using a BrainComputer Interface (BCI) that interacts with the eMotiv EPOC Neural Headset. The contribution of the work is an alternative input device for those who have a motor disability and are challenged by traditional input devices. The advantages of a virtual keyboard based on BCI are summarized and we describe its design and implementation. We also present the results of a preliminary study that has suggested several improvements for enhancing the effectiveness of the virtual keyboard.

**REQUIREMENTS SPECIFICATION**

**CHAPTER 3**

**REQUIREMENTS SPECIFICATION**

**3.1 Hardware Requirements**

|  |  |
| --- | --- |
| Processor | 2.0 GHz or higher |
| Storage | Minimum 80 GB |
| Memory | Minimum 8 GB |
| Output Units | Monitor |
| Input Units | Keyboard, Headset, Internet |

**3.2 Software Requirements**

|  |  |
| --- | --- |
| Operating System | Linux or Windows or Mac |
| Programming Packages | Python Packages, Emotiv Cortex API |
| Softwares | PyCharm, Web browser, Github |

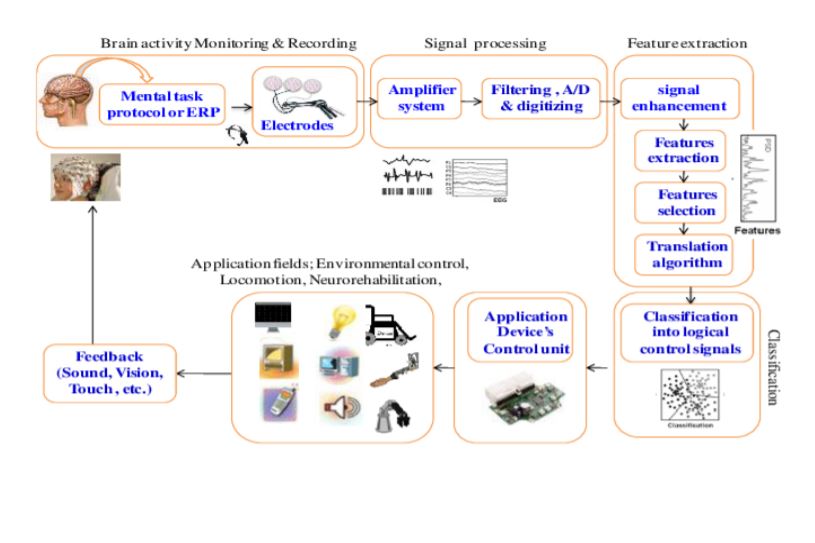
**SYSTEM DESIGN AND ARCHITECTURE**

**CHAPTER 4**

**SYSTEM DESIGN AND ARCHITECTURE**

**4.1 Design**

**4.1.1 System Architecture**

****

**Figure 1. Block Diagram of System Architecture**

**COMPONENTS OF DFD DIAGRAM:**

**1.Common Average Reference(CAR):** common average reference minimizes uncorrelated sources of signal and noise through averaging, while eliminating sources of noise common to all sites. Therefore a common average reference more closely approximates the theoretical differential recording idea.

**2. Bandpass Filters**: A bandpass filter is an electronic device or circuit that allows signals between two specific frequencies to pass, but that discriminates against signals at other frequencies. 3.Common Spatial Filters(CSP): The CSP (Common Spatial Patterns) has been proved to be an effective feature extraction method in Brain Computer Interfaces. It is widely used for two-class problem.The obtained features are input to SVM (Support Vector Machine) for classification.

**3.SVM Classifier:** A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples.

**4.Artificial Subspace Reconstruction(ASR):** Used to remove high amplitude artifects ex:eye blinks,muscle burst.

**5.Gait Cycle Segments:** is the interval between two successive hell stride incidences.

**6.IC clusters:** Cluster-mean scalp maps for the seven independent component (IC) clusters and their IC equivalent dipoles. Cluster centroid dipole locations.

**7.Independent Component Analysis(ICA):** (ICA) is a statistical and computational technique for revealing hidden factors that underlie sets of random variables, measurements, or signals. ICA defines a generative model for the observed multivariate data, which is typically given as a large database of samples.

A screenshot of a cell phone

Description generated with very high confidence

**Figure 2. Setup Tab**

# Technical Specifications of Emotiv insight

​

|  |  |  |
| --- | --- | --- |
| **Headset Version** | Pre-2019 | 2019 |
| **Number of Channels** | 5 (plus CMS/DRL reference on left mastoid) | 5 (plus CMS/DRL reference on left mastoid) |
| **Channel names (International 10-20 locations)** | AF3, AF4, T7, T8, Pz | AF3, AF4, T7, T8, Pz |
| **Sampling Method** | Sequential sampling, single ADC | Sequential sampling, single ADC |
| **Sampling Rate** | 128 SPS (2048 Hz internal) | 128 SPS (2048 Hz internal) |
| **EEG Resolution** | 14 bits 1 LSB = 0.51μV (16 bit ADC, 2 bits instrumental noise floor discarded) | 14 bits 1 LSB = 0.51μV (16 bit ADC, 2 bits instrumental noise floor discarded) |
| **Bandwidth** | 0.5 - 43Hz, digital notch filters at 50Hz and 60Hz | 0.5 - 43Hz, digital notch filters at 50Hz and 60Hz |
| **Filtering** | Built in digital 5th order Sinc filter | Built in digital 5th order Sinc filter |
| **Dynamic Range (Input referred)** | 8400 uV (pp) | 8400 uV (pp) |
| **Coupling Mode** | AC coupled | AC coupled |
| **Connectivity** | Proprietary 2.4GHz wireless (with dongle), BLE | Proprietary 2.4GHz wireless (with dongle), BLE |
| **Battery Capacity** | LiPo battery 480mAh | LiPo battery 450mAh |
| **Battery Life (typical)** | Up to 8 hours | Up to 8 hours |
| **Impedance Measurement** | Real-time contact quality using patented system | Real-time contact quality using patented system |
| **IMU Part** | LSM9DS0 | ICM-20948 |
| **Accelerometer** | 3-axis +/-8g | 3-axis +/-4g |
| **Gyroscope** | 3-axis +/- 500 dps | Output as Quaternion |
| **Magnetometer** | 3-axis +/- 12 gauss | 3-axis +/- 4900 uTesla |
| **Motion Sampling** | 128 Hz | 64 Hz |
| **Quaternion Outputs** | No | Yes |
| **Motion Resolution** | 14-bit | 14-bit (dongle) / 8-bit (BLE) |
| **Sensor Material** | Semi dry polymer | Semi dry polymer |
|  | **Figure 3. Technical Specifications** |  |

**4.2 Flowchart**

A close up of text on a white background

Description generated with high confidence

**Figure 5. Flowchart of API flow**

**4.3 Modules**

**4.3.1 Cortex API**

Cortex API is the the core piece of technology at EMOTIV which brings the brain computer interface to consumer. The Cortex API is built on JSON and WebSockets, making it easy to access from a variety of programming languages and platforms.

There are a few steps to prepare before you can get the data streaming out from Cortex:

1. First you have to login to the EMOTIV App with your [EmotivID](file:///C:\cortex-api\#create-an-emotivid).
2. Make sure that you are using the correct application ID, client ID and client secret that you generated from the Account dashboard on Emotiv website in your application.
3. Call the [requestAccess](file:///C:\cortex-api\authentication\requestaccess) API from your app to Cortex, and accept via the EMOTIV App.
4. After access is granted, connect your EMOTIV brainwear headset via the USB dongle or Bluetooth.
5. Call the [queryHeadsets](file:///C:\cortex-api\headset\queryheadsets) API to list the available headsets.
6. Call the [controlDevice](file:///C:\cortex-api\headset\controldevice) API to connect to the desire headset.
7. Call the [authorize](file:///C:\cortex-api\authentication\authorize) API to get a Cortex token for subsequence requests.
8. Call the [createSession](file:///C:\cortex-api\session\createsession) API to open up a new session and be ready for BCI data streaming.

**WebSocket:-**

Communicate with the Cortex API using the [WebSocket Secure](https://en.wikipedia.org/wiki/WebSocket) protocol.

Create a **WebSocket client** and connect to **localhost** on the port **6868**, using the **wss** protocol.

Any WebSocket client should work in any programming language. You can even use a web browser plugin or an [online client](https://www.websocket.org/echo.html).

**JSON-RPC:-**

After you have successfully opened the WebSocket connection, communicate with Cortex using the [JSON-RPC 2.0](https://www.jsonrpc.org/specification) protocol.

Call **methods**, with or without **parameters**, and Cortex sends back a **result** or an **error**.

# Authentication

After your application is successfully [connected](/cortex-api/connecting-to-the-cortex-api) to the Cortex service, you must go through the authentication procedure.

First, you should call [getUserLogin](/cortex-api/authentication/getuserlogin) to check if the user has already logged in though EMOTIV App. Then, you must call [requestAccess](/cortex-api/authentication/requestaccess) to ask the user to approve your application.

Finally, call [authorize](/cortex-api/authentication/authorize) to generate a Cortex token or you can reuse a token that you previously got from this method, if it is not expired.

# requestAccess

Request user approval for the current application through EMOTIV App.

When your application calls this method for the first time, EMOTIV App displays a message to approve your application. You can call this API many times, but EMOTIV App will prompt the user only once. If the user has already approved your application, then this API does nothing.

Almost all the methods of the API require that your application was approved in EMOTIV App. These methods will return an error if your application wasn't approved.

# authorize

This method is to generate a Cortex access token. Most of the methods of the Cortex API require this token as a parameter.

Application can specify the license key and the amount of sessions to be debited from the license and use them locally.

The token is linked to your application. It cannot be used with another application. The token is also linked to the EmotivID of the current user. It cannot be used with another EmotivID. So if the user logs out in EMOTIV App, and then logs in with another EmotivID, your application must call this API again to get a new token.

Your application can save the Cortex token and reuse it later, within 2 days. Note that it is the responsibility of the application to secure the token.

If the user has not accepted the EULA, then a warning message will be included in the response as well. The user must accept the EULA through EMOTIV App.

# Headsets

After you finish the [authentication](/cortex-api/authentication) process, your application should find an EMOTIV headset to work with, using the method [queryHeadsets](/cortex-api/headset/queryheadsets).

If the headset is not connected to Cortex yet, then you must call [controlDevice](/cortex-api/headset/controldevice).

# Sessions

A session is an object that makes the link between your application and an EMOTIV headset. When the user wants to work with a headset, your application should create a session first. Then you can:

* subscribe to the [data stream](https://emotiv.gitbook.io/cortex-api/data-subscription) of the headset
* create a [record](https://emotiv.gitbook.io/cortex-api/records) and add [markers](https://emotiv.gitbook.io/cortex-api/markers)​
* use [BCI](https://emotiv.gitbook.io/cortex-api/bci)​

Your application can open only one session at a time with a given headset. But it can open multiple sessions with multiple headsets

**IMPLEMENTATION**

**CHAPTER 5**

**IMPLEMENTATION**

**5.1 Languages and Platform Used For Implementation**

**5.1.1 Python**

Python programming language supports the usage of modules and packages; this means that the program can be developed in a modular way and code can be reused in various projects. Python language is easily understandable and readable. It is user-friendly and high-level programming language and also easy to learn and use CPython is the reference implementation of Python, written in C. It compiles Python code to intermediate bytecode which is then interpreted by a virtual machine. All versions of the Python language are implemented in C because CPython is the reference implementation.

According to the latest TIOBE Programming Community Index, Python is one of the top 10 popular programming languages of 2017. Python is a general purpose and high level programming language. You can use Python for developing desktop GUI applications, websites and web applications. Also, Python, as a high level programming language, allows you to focus on core functionality of the application by taking care of common programming tasks. The simple syntax rules of the programming language further makes it easier for you to keep the code base readable and application maintainable. There are also a number of reasons why you should prefer Python to other programming languages.

5.1.b. Features of Python

* Easy to Learn and Use. Python is easy to learn and use. ...
* Expressive Language. ...
* Interpreted Language. ...
* Cross-platform Language. ...
* Free and Open Source. ...
* Object-Oriented Language. ...
* Extensible. ...
* Large Standard Library.

5.1.a. Advantages

* Presence of Third Party Modules: ...
* Extensive Support Libraries: ...
* Open Source and Community Development: ...
* Learning Ease and Support Available: ...
* User-friendly Data Structures: ...
* Productivity and Speed

There are a few steps to prepare before you can get the data streaming out from Cortex:

1. First you have to login to the EMOTIV App with your [EmotivID](file:///C:\cortex-api\#create-an-emotivid).
2. Make sure that you are using the correct application ID, client ID and client secret that you generated from the Account dashboard on Emotiv website in your application.
3. Call the [requestAccess](file:///C:\cortex-api\authentication\requestaccess) API from your app to Cortex, and accept via the EMOTIV App.
4. After access is granted, connect your EMOTIV brainwear headset via the USB dongle or Bluetooth.
5. Call the [queryHeadsets](file:///C:\cortex-api\headset\queryheadsets) API to list the available headsets.
6. Call the [controlDevice](file:///C:\cortex-api\headset\controldevice) API to connect to the desire headset.
7. Call the [authorize](file:///C:\cortex-api\authentication\authorize) API to get a Cortex token for subsequence requests.
8. Call the [createSession](file:///C:\cortex-api\session\createsession) API to open up a new session and be ready for BCI data streaming.

**5.2 Software Used For Implementation**

**5.2.1 PyCharm**

PyCharm is a Python IDE with complete set of tools for Python development. In addition, the IDE provides capabilities for professional Web development using the Django framework. Code faster and with more easily in a smart and configurable editor with code completion, snippets, code folding and split windows support.

**PyCharm Features**

* **Intelligent Coding Assistance** - PyCharm provides smart code completion, code inspections, on-the-fly error highlighting and quick-fixes, along with automated code refactorings and rich navigation capabilities.
* **Intelligent Code Editor** - PyCharm’s smart code editor provides first-class support for Python, JavaScript, CoffeeScript, TypeScript, CSS, popular template languages and more. Take advantage of language-aware code completion, error detection, and on-the-fly code fixes!
* **Smart Code Navigation** - Use smart search to jump to any class, file or symbol, or even any IDE action or tool window. It only takes one click to switch to the declaration, super method, test, usages, implementation, and more.
* **Fast and Safe Refactorings** - Refactor your code the intelligent way, with safe Rename and Delete, Extract Method, Introduce Variable, Inline Variable or Method, and other refactorings. Language and framework-specific refactorings help you perform project-wide changes.
* **Built-in Developer Tools** - PyCharm’s huge collection of tools out of the box includes an integrated debugger and test runner; Python profiler; a built-in terminal; integration with major VCS and built-in database tools; remote development capabilities with remote interpreters; an integrated ssh terminal; and integration with Docker and Vagrant.
* **Debugging, Testing and Profiling** - Use the powerful debugger with a graphical UI for Python and JavaScript. Create and run your tests with coding assistance and a GUI-based test runner. Take full control of your code with Python Profiler integration.
* **VCS, Deployment and Remote Development** - Save time with a unified UI for working with Git, SVN, Mercurial or other version control systems. Run and debug your application on remote machines. Easily configure automatic deployment to a remote host or VM and manage your infrastructure with Vagrant and Docker.
* **Database tools** - Access Oracle, SQL Server, PostgreSQL, MySQL and other databases right from the IDE. Rely on PyCharm’s help when editing SQL code, running queries, browsing data, and altering schemas.
* **Web Development** - In addition to Python, PyCharm provides first-class support for various Python web development frameworks, specific template languages, JavaScript, CoffeeScript, TypeScript, HTML/CSS, AngularJS, Node.js, and more.
* **Python Web frameworks** - PyCharm offers great framework-specific support for modern web development frameworks such as Django, Flask, Google App Engine, Pyramid, and web2py, including Django templates debugger, manage.py and appcfg.py tools, special autocompletion and navigation, just to name a few.
* **JavaScript & HTML -**PyCharm provides first-class support for JavaScript, CoffeeScript, TypeScript, HTML and CSS, as well as their modern successors. The JavaScript debugger is included in PyCharm and is integrated with the Django server run configuration.
* **Live Edit** - Live Editing Preview lets you open a page in the editor and the browser and see the changes being made in code instantly in the browser. PyCharm auto-saves your changes, and the browser smartly updates the page on the fly, showing your edits.
* **Scientific Tools** - PyCharm integrates with IPython Notebook, has an interactive Python console, and supports Anaconda as well as multiple scientific packages including Matplotlib and NumPy.
* **Interactive Python console** - You can run a REPL Python console in PyCharm which offers many advantages over the standard one: on-the-fly syntax check with inspections, braces and quotes matching, and of course code completion.
* **Scientific Stack Support** - PyCharm has built-in support for scientific libraries. It supports Pandas, Numpy, Matplotlib, and other scientific libraries, offering you best-in-class code intelligence, graphs, array viewers and much more.
* **Conda Integration** - Keep your dependencies isolated by having separate Conda environments per project, PyCharm makes it easy for you to create and select the right environment.
* **Customizable and Cross-platform IDE** - Use PyCharm on Windows, Mac OS and Linux with a single license key. Enjoy a fine-tuned workspace with customizable color schemes and key-bindings, with VIM emulation available.
* **Customizable UI** - Enjoy a fine-tuned workspace with customizable color schemes and key-bindings.
* **Plugins** - More than 10 years of IntelliJ platform development gives PyCharm 50+ IDE plugins of different nature, including support for additional VCS, integrations with different tools and frameworks, and editor enhancements such as Vim emulation.
* **Cross-platform IDE** - PyCharm works on Windows, Mac OS or Linux. You can install and run PyCharm on as many machines as you have, and use the same environment and functionality across all your machines.

**5.3 Code**

**5.3.1 Implementation code**

Since Python is main development language used, all the files are executed using .py environment. Accordingly the following node packages are installed.

1. tkinter
2. json
3. Pillow
4. Itertools
5. Websocket

**Connector.py**

import json

import ssl

import time

from itertools import cycle

from tkinter import \*

import websocket

class Cortex():

def \_\_init\_\_(self, url, user):

self.ws = websocket.create\_connection(url, sslopt={"cert\_reqs": ssl.CERT\_NONE})

self.user = user

self.packet\_count = 0

self.count = 0

self.id\_sequence = 0

self.switch\_on\_off=None

def query\_headset(self):

QUERY\_HEADSET\_ID = 2

query\_headset\_request = {

"jsonrpc": "2.0",

"id": QUERY\_HEADSET\_ID,

"method": "queryHeadsets",

"params": {}

}

self.ws.send(json.dumps(query\_headset\_request))

result = self.ws.recv()

result\_dic = json.loads(result)

if result\_dic['result'] == []:

print("connect your device")

# print('query headset result', json.dumps(result\_dic, indent=4))

self.headset\_id = result\_dic['result'][0]['id']

print(self.headset\_id)

def connect\_headset(self):

CONNECT\_HEADSET\_ID = 111

connect\_headset\_request = {

"jsonrpc": "2.0",

"id": CONNECT\_HEADSET\_ID,

"method": "controlDevice",

"params": {

"command": "connect",

"headset": self.headset\_id

}

}

self.ws.send(json.dumps(connect\_headset\_request))

result = self.ws.recv()

result\_dic = json.loads(result)

print('connect headset result', json.dumps(result\_dic, indent=4))

def request\_access(self):

REQUEST\_ACCESS\_ID = 1

request\_access\_request = {

"jsonrpc": "2.0",

"method": "requestAccess",

"params": {

"clientId": self.user['client\_id'],

"clientSecret": self.user['client\_secret']

},

"id": REQUEST\_ACCESS\_ID

}

self.ws.send(json.dumps(request\_access\_request))

result = self.ws.recv()

result\_dic = json.loads(result)

print(json.dumps(result\_dic, indent=4))

def authorize(self):

AUTHORIZE\_ID = 4

authorize\_request = {

"jsonrpc": "2.0",

"method": "authorize",

"params": {

"clientId": self.user['client\_id'],

"clientSecret": self.user['client\_secret'],

"license": self.user['license'],

"debit": self.user['debit']

},

"id": AUTHORIZE\_ID

}

print('json.dumps(authorize\_request)', json.dumps(authorize\_request))

self.ws.send(json.dumps(authorize\_request))

result = self.ws.recv()

result\_dic = json.loads(result)

print('auth\_result', json.dumps(result\_dic, indent=4))

self.auth = result\_dic['result']['cortexToken']

print(self.auth)

def create\_session(self, auth, headset\_id):

CREATE\_SESSION\_ID = 5

try:

create\_session\_request = {

"jsonrpc": "2.0",

"id": CREATE\_SESSION\_ID,

"method": "createSession",

"params": {

"cortexToken": self.auth,

"headset": self.headset\_id,

"status": "active"

}

}

self.ws.send(json.dumps(create\_session\_request))

result = self.ws.recv()

result\_dic = json.loads(result)

print('create session result ', json.dumps(result\_dic, indent=4))

self.session\_id = result\_dic['result']['id']

#print(self.session\_id)

except KeyError:

error="error"

return error

def close\_session(self):

CREATE\_SESSION\_ID = 117

close\_session\_request = {

"jsonrpc": "2.0",

"id": CREATE\_SESSION\_ID,

"method": "updateSession",

"params": {

"cortexToken": self.auth,

"session": self.session\_id,

"status": "close"

}

}

self.ws.send(json.dumps(close\_session\_request))

result = self.ws.recv()

result\_dic = json.loads(result)

print('close session result ', json.dumps(result\_dic, indent=4))

#def get\_cortex\_info(self):

# get\_cortex\_info\_request = {

# "jsonrpc": "2.0",

# "method": "getCortexInfo",

# "id": 100

# }

# self.ws.send(json.dumps(get\_cortex\_info\_request))

#result = self.ws.recv()

#print(json.dumps(json.loads(result), indent=4))

def grant\_access\_and\_session\_info(self):

self.query\_headset()

self.connect\_headset()

self.request\_access()

self.authorize()

self.create\_session(self.auth, self.headset\_id)

def gen\_request(self, method, auth, \*\*kwargs):

self.id\_sequence += 1

params = {key: value for (key, value) in kwargs.items()}

if auth and self.auth:

params['cortexToken'] = self.auth

request = json.dumps(

{'jsonrpc': "2.0",

'method': method,

'params': params,

'id': self.id\_sequence

})

print(f"Sending request:\n{request}")

return request

def send\_command(self, method, auth=True, callback=None, \*\*kwargs):

if auth and not self.auth:

self.authorize()

msg = self.gen\_request(method, auth, \*\*kwargs)

self.ws.send(msg)

print("sent; awaiting response")

resp = self.ws.recv()

if 'error' in resp:

print(f"Got error in {method} with params {kwargs}:\n{resp}")

raise Exception(resp)

resp = json.loads(resp)

if callback:

callback(resp)

return resp

def disconnect\_headset(self):

DISCONNECT\_HEADSET\_ID = 112

disconnect\_headset\_request = {

"jsonrpc": "2.0",

"id": DISCONNECT\_HEADSET\_ID,

"method": "controlDevice",

"params": {

"command": "disconnect",

"headset": self.headset\_id

}

}

self.ws.send(json.dumps(disconnect\_headset\_request))

# wait until disconnect completed

while True:

time.sleep(1)

result = self.ws.recv()

result\_dic = json.loads(result)

print('disconnect headset result', json.dumps(result\_dic, indent=4))

if 'warning' in result\_dic:

if result\_dic['warning']['code'] == 1:

break

def subscribe(self, stream\_list):

params = {'cortexToken': self.auth,

'session': self.session\_id,

'streams': stream\_list}

resp = self.send\_command('subscribe', \*\*params)

print(f"{\_\_name\_\_} resp:\n{resp}")

def get\_data(self):

while True:

resp = self.ws.recv()

res = json.loads(resp)

val = res['fac']

eyeact = val[0]

alternator = cycle(("on", "off"))

if eyeact == 'blink':

self.count += 1

if self.count == 14:

self.switch\_on\_off=next(alternator)

print(self.switch\_on\_off)

#time.sleep(0.5)

if self.count == 24:

self.switch\_on\_off="off"

print(self.switch\_on\_off)

self.count = 0

#time.sleep(0.5)

self.packet\_count += 1

def queryProfile(self):

QUERY\_PROFILE\_ID = 15

queryProfileRequest = {

"jsonrpc": "2.0",

"method": "queryProfile",

"params": {

"cortexToken": self.auth,

},

"id": QUERY\_PROFILE\_ID

}

print('query profile:\n', json.dumps(queryProfileRequest))

print('\n')

self.ws.send(json.dumps(queryProfileRequest))

result = self.ws.recv()

result\_dic = json.loads(result)

self.queryProfile = result\_dic

print('result queryProfile\n', result\_dic)

print('\n')

**Cortex.py**

import json

import ssl

import time

from itertools import cycle

from tkinter import \*

import websocket

class Cortex():

def \_\_init\_\_(self, url, user):

self.ws = websocket.create\_connection(url, sslopt={"cert\_reqs": ssl.CERT\_NONE})

self.user = user

self.packet\_count = 0

self.count = 0

self.id\_sequence = 0

self.switch\_on\_off=None

def query\_headset(self):

QUERY\_HEADSET\_ID = 2

query\_headset\_request = {

"jsonrpc": "2.0",

"id": QUERY\_HEADSET\_ID,

"method": "queryHeadsets",

"params": {}

}

self.ws.send(json.dumps(query\_headset\_request))

result = self.ws.recv()

result\_dic = json.loads(result)

if result\_dic['result'] == []:

print("connect your device")

# print('query headset result', json.dumps(result\_dic, indent=4))

self.headset\_id = result\_dic['result'][0]['id']

return self.headset\_id

def connect\_headset(self):

CONNECT\_HEADSET\_ID = 111

connect\_headset\_request = {

"jsonrpc": "2.0",

"id": CONNECT\_HEADSET\_ID,

"method": "controlDevice",

"params": {

"command": "connect",

"headset": self.headset\_id

}

}

self.ws.send(json.dumps(connect\_headset\_request))

result = self.ws.recv()

result\_dic = json.loads(result)

print('connect headset result', json.dumps(result\_dic, indent=4))

def request\_access(self):

REQUEST\_ACCESS\_ID = 1

request\_access\_request = {

"jsonrpc": "2.0",

"method": "requestAccess",

"params": {

"clientId": self.user['client\_id'],

"clientSecret": self.user['client\_secret']

},

"id": REQUEST\_ACCESS\_ID

}

self.ws.send(json.dumps(request\_access\_request))

result = self.ws.recv()

result\_dic = json.loads(result)

print(json.dumps(result\_dic, indent=4))

def authorize(self):

AUTHORIZE\_ID = 4

authorize\_request = {

"jsonrpc": "2.0",

"method": "authorize",

"params": {

"clientId": self.user['client\_id'],

"clientSecret": self.user['client\_secret'],

"license": self.user['license'],

"debit": self.user['debit']

},

"id": AUTHORIZE\_ID

}

print('json.dumps(authorize\_request)', json.dumps(authorize\_request))

self.ws.send(json.dumps(authorize\_request))

result = self.ws.recv()

result\_dic = json.loads(result)

print('auth\_result', json.dumps(result\_dic, indent=4))

self.auth = result\_dic['result']['cortexToken']

print(self.auth)

def create\_session(self, auth, headset\_id):

CREATE\_SESSION\_ID = 5

create\_session\_request = {

"jsonrpc": "2.0",

"id": CREATE\_SESSION\_ID,

"method": "createSession",

"params": {

"cortexToken": self.auth,

"headset": self.headset\_id,

"status": "active"

}

}

self.ws.send(json.dumps(create\_session\_request))

result = self.ws.recv()

result\_dic = json.loads(result)

print('create session result ', json.dumps(result\_dic, indent=4))

self.session\_id = result\_dic['result']['id']

# print(self.session\_id)

def close\_session(self):

CREATE\_SESSION\_ID = 117

close\_session\_request = {

"jsonrpc": "2.0",

"id": CREATE\_SESSION\_ID,

"method": "updateSession",

"params": {

"cortexToken": self.auth,

"session": self.session\_id,

"status": "close"

}

}

self.ws.send(json.dumps(close\_session\_request))

result = self.ws.recv()

result\_dic = json.loads(result)

print('close session result ', json.dumps(result\_dic, indent=4))

#def get\_cortex\_info(self):

# get\_cortex\_info\_request = {

# "jsonrpc": "2.0",

# "method": "getCortexInfo",

# "id": 100

# }

# self.ws.send(json.dumps(get\_cortex\_info\_request))

#result = self.ws.recv()

#print(json.dumps(json.loads(result), indent=4))

def grant\_access\_and\_session\_info(self):

self.query\_headset()

self.connect\_headset()

self.request\_access()

self.authorize()

self.create\_session(self.auth, self.headset\_id)

def gen\_request(self, method, auth, \*\*kwargs):

self.id\_sequence += 1

params = {key: value for (key, value) in kwargs.items()}

if auth and self.auth:

params['cortexToken'] = self.auth

request = json.dumps(

{'jsonrpc': "2.0",

'method': method,

'params': params,

'id': self.id\_sequence

})

print(f"Sending request:\n{request}")

return request

def send\_command(self, method, auth=True, callback=None, \*\*kwargs):

if auth and not self.auth:

self.authorize()

msg = self.gen\_request(method, auth, \*\*kwargs)

self.ws.send(msg)

print("sent; awaiting response")

resp = self.ws.recv()

if 'error' in resp:

print(f"Got error in {method} with params {kwargs}:\n{resp}")

raise Exception(resp)

resp = json.loads(resp)

if callback:

callback(resp)

return resp

def setup\_profile(self, profile\_name, status):

print('setup profile --------------------------------')

setup\_profile\_json = {

"jsonrpc": "2.0",

"method": "setupProfile",

"params": {

"cortexToken": self.auth,

"headset": self.headset\_id,

"profile": profile\_name,

"status": status

},

}

self.ws.send(json.dumps(setup\_profile\_json))

result = self.ws.recv()

result\_dic = json.loads(result)

print('result \n', json.dumps(result\_dic, indent=4))

print('\n')

def disconnect\_headset(self):

DISCONNECT\_HEADSET\_ID = 112

disconnect\_headset\_request = {

"jsonrpc": "2.0",

"id": DISCONNECT\_HEADSET\_ID,

"method": "controlDevice",

"params": {

"command": "disconnect",

"headset": self.headset\_id

}

}

self.ws.send(json.dumps(disconnect\_headset\_request))

# wait until disconnect completed

while True:

time.sleep(1)

result = self.ws.recv()

result\_dic = json.loads(result)

print('disconnect headset result', json.dumps(result\_dic, indent=4))

if 'warning' in result\_dic:

if result\_dic['warning']['code'] == 1:

break

def subscribe(self, stream\_list):

params = {'cortexToken': self.auth,

'session': self.session\_id,

'streams': stream\_list}

resp = self.send\_command('subscribe', \*\*params)

print(f"{\_\_name\_\_} resp:\n{resp}")

def get\_data(self):

while True:

resp = self.ws.recv()

res = json.loads(resp)

val = res['fac']

eyeact = val[0]

alternator = cycle(("on", "off"))

if eyeact == 'blink':

self.count += 1

if self.count == 14:

self.switch\_on\_off=next(alternator)

print(self.switch\_on\_off)

#time.sleep(0.5)

if self.count == 24:

self.switch\_on\_off="off"

print(self.switch\_on\_off)

self.count = 0

#time.sleep(0.5)

self.packet\_count += 1

def queryProfile(self):

QUERY\_PROFILE\_ID = 15

queryProfileRequest = {

"jsonrpc": "2.0",

"method": "queryProfile",

"params": {

"cortexToken": self.auth,

},

"id": QUERY\_PROFILE\_ID

}

print('query profile:\n', json.dumps(queryProfileRequest))

print('\n')

self.ws.send(json.dumps(queryProfileRequest))

result = self.ws.recv()

result\_dic = json.loads(result)

self.queryProfile = result\_dic

print('result queryProfile\n', result\_dic)

print('\n')

**MainNavigator.py**

from tkinter import \*

from tkinter import messagebox

import PIL.Image

from PIL import ImageTk

from connector import \*

from IOT import eyeblink

from facial\_expression\_Train import FacialExpression

from Cursor\_Train import CursorTrainer

from facialExpressionLive import FacialExpressionLive

from ActSpeak import Act2Speak

from Cursor\_live\_mode import CursorLive

class SampleApp(Tk):

def \_\_init\_\_(self, \*args, \*\*kwargs):

Tk.\_\_init\_\_(self, \*args, \*\*kwargs)

self.wm\_iconbitmap('logo.ico')

self.title('BCOT')

w, h = self.winfo\_screenwidth(), self.winfo\_screenheight()

self.geometry("%dx%d+0+0" % (w, h))

self.container = Frame(self)

self.container.pack(side="top", fill="both", expand=True)

self.container.grid\_rowconfigure(0, weight=1)

self.container.grid\_columnconfigure(0, weight=1)

self.frames = {}

self.frames = {

"StartPage": StartPage,

"Navigator":Navigator,

"eyeblink":eyeblink,

"FacialExpression":FacialExpression,

"CursorTrainer":CursorTrainer,

"FacialExpressionLive":FacialExpressionLive,

"Act2Speak":Act2Speak,

"CursorLive":CursorLive

}

self.show\_frame("FacialExpression")

def show\_frame(self, page\_name):

# destroy the old frame

for child in self.container.winfo\_children():

child.destroy()

if page\_name == 'eyeblink':

#frame\_class = self.frames[page\_name]

self.after(7000,self.show\_frame,"Navigator")

# create the new frame

frame\_class = self.frames[page\_name]

frame = frame\_class(parent=self.container, controller=self)

frame.pack(fill="both", expand=True)

class StartPage(Frame):

def \_\_init\_\_(self, parent, controller):

Frame.\_\_init\_\_(self, parent)

self.controller=controller

canvas = Canvas(self,width=2085, height=1080, bg='#020A2E')

canvas.pack()

self.img = PIL.Image.open("images/b100.png")

self.my\_img = ImageTk.PhotoImage(self.img)

canvas.create\_image(200, 310, image=self.my\_img, tags='close\_tag')

self.imgg = PIL.Image.open("images/c100.png")

self.myimg = ImageTk.PhotoImage(self.imgg)

canvas.create\_image(470, 310, image=self.myimg)

self.oimage = PIL.Image.open("images/o100.png")

self.omg = ImageTk.PhotoImage(self.oimage)

canvas.create\_image(820, 310, image=self.omg)

self.timg = PIL.Image.open("images/t100.png")

self.tmg = ImageTk.PhotoImage(self.timg)

canvas.create\_image(1150, 310, image=self.tmg)

canvas.create\_text(665, 540, text="connect to headset", anchor=S, font='Arial 8 ', fill='white')

self.url = "wss://localhost:6868"

self.user = {

"license": "1d10f0b4-65e4-4424-ae8a-a56ae2fa6950",

"client\_id": "uNU5UMKd9eFLYp8JtHr6ZXXaLHg1p6rf1BReVn4N",

"client\_secret": "9wcDtQn2Wubjg7zzlO2tnkx8Hzk1GkLpqZsqgOiuaWsaI0VRkxcxeQ9ZOPrHrNvJ1tlgOAV1XEZ6ooxJ06sRwobXDApRsol08w9YsJWU0fVAieWYp6kHexOlM9OWqXWk",

"debit": 100

# "number\_row\_data" : 10

}

connect = Button(self,width=15, height=1, bd=0, bg='#2C3547', text="CONNECT", relief=RAISED,

font='Arial 10 bold ', fg='white', command=self.connection).place(x=600, y=500)

self.next = PIL.Image.open("images/next.png")

self.img\_nxt = ImageTk.PhotoImage(self.next)

Button(width=35, height=30, bg='#020A2E', image=self.img\_nxt, bd=0, command=lambda: controller.show\_frame('Navigator')).place(x=1320, y=10)

def connection(self):

c = Cortex(self.url, self.user)

headset\_id = c.query\_headset()

c.connect\_headset()

c.request\_access()

auth = c.authorize()

val=c.create\_session(auth, headset\_id)

if val == "error":

messagebox.showinfo("BCOT","headset is not connected")

else:

messagebox.showinfo("BCOT","headset is connected")

class Navigator(Frame):

def \_\_init\_\_(self,parent,controller):

Frame.\_\_init\_\_(self,parent)

self.controller=controller

self.image1 = PhotoImage(file=r'images/iot.png')

self.image2 = PhotoImage(file=r'images/facialExp.png')

self.image3 = PhotoImage(file=r'images/cursor.png')

self.image4 = PhotoImage(file=r'images/mindR.png')

self.imgg = PhotoImage(file=r'images/arrow.png')

self.controller = controller

self.canvas = Canvas(self,width=2085, height=1080, bg='#020A2E')

self.canvas.pack()

label = Label(self,text="FEATURES", bg='#020A2E', fg='white', font='Arial 50 bold').place(x=80, y=20)

arrow = Button(self,width=40, height=30, bg='#020A2E', image=self.imgg, bd=0,

command=lambda:controller.show\_frame("StartPage")).place(x=10, y=10)

button1 =Button(self,width=200, height=215, bg="#3A3535", bd=0, image=self.image1,command=lambda:controller.show\_frame("eyeblink")).place(x=380, y=150)

self.canvas.create\_rectangle(75, 150, 380, 365, fill='#615A5A')

self.canvas.create\_text(220, 160, text="BLINK FOR ON/OFF", anchor=N, font='Arial 14 bold', fill='white')

self.canvas.create\_text(223,230,text=" Feature contains IOT Devices-\n"

" Focus over the icon and blink twice\n"

" to turn on/off.",font='Arial 12 bold',fill='white')

self.canvas.create\_text(220, 270, text=" This feature will be active for 7 seconds and the\n"

" frame will automatically closes by itself, after \n"

" the frame is opened wait for 3 seconds to, then \n"

" start to blink.", anchor=N, font='Arial 9 bold', fill='white')

button2 = Button(self,width=200, height=215, bg="#3A3535", bd=0, image=self.image2,command=lambda:controller.show\_frame("FacialExpression")).place(x=1100, y=150)

self.canvas.create\_rectangle(795, 150, 1101, 368, fill='#615A5A')

self.canvas.create\_text(950, 160, text="FACIAL EXPRESSION", anchor=N, font='Arial 14 bold', fill='white')

# canvas.create\_rectangle(770, 130,1325,390, fill='#FFFFFF')

self.canvas.create\_text(948, 230, text="This feature has Training Frame and\n"

" Live Frame-The more you train more\n"

" accuracy obtained.", font='Arial 12 bold', fill='white')

self.canvas.create\_text(945, 302, text=" The training frame has 5 facial commands that\n"

" needs to be trained,if the live feed is not accurate\n"

" then train neutral more. The live feed takes 3 sec\n"

" to initiate connection."

, font='Arial 9 bold', fill='white')

button3 = Button(self,width=200, height=215, bg="#3A3535", bd=0, image=self.image3,command=lambda:controller.show\_frame("CursorTrainer")).place(x=380, y=450)

self.canvas.create\_rectangle(75, 450, 380, 667, fill='#615A5A')

self.canvas.create\_text(220, 460, text="CURSOR CONTROL", anchor=N, font='Arial 14 bold', fill='white')

# canvas.create\_rectangle(50,430,607,690,fill='#FFFFFF')

self.canvas.create\_text(223, 530, text=" The feature has 4 mental commands\n"

" to train and live mode frame allows\n"

" to control the whole application.", font='Arial 12 bold', fill='white')

self.canvas.create\_text(223, 600, text=" The training frame has 4 mental commands\n"

" that needs to be trained, if the live feed is not\n"

" accurate then train neutral more. The live feed \n"

" takes 3 sec to initiate connection.", font='Arial 9 bold',fill='white')

button4 = Button(self,width=200, height=215, bg="#3A3535", bd=0, image=self.image4,command=lambda:controller.show\_frame("Act2Speak")).place(x=1100, y=450)

self.canvas.create\_rectangle(795, 450, 1100, 669, fill='#615A5A')

self.canvas.create\_text(950, 460, text="ACT TO SPEECH", anchor=N, font='Arial 14 bold', fill='white')

# canvas.create\_rectangle(770,430,1325,688,fill='#FFFFFF')

self.canvas.create\_text(948, 530, text="This feature has an Artificial voice -\n"

" speaks when muscular movement is\n"

" made or thought about something.", font='Arial 12 bold', fill='white')

self.canvas.create\_text(948, 600, text="This allows person to make 8 different speechs\n"

"the number can be increased but it takes more\n"

"cognitive ability.",font='Arial 9 bold',fill='white')

self.my\_img = PhotoImage(file="images/sky.png")

self.my\_rectangle = self.canvas.create\_image(330, 255, image=self.my\_img, tags='close\_tag')

# my\_rectangle=canvas.create\_rectangle(50,130,610,390,fill="#FFFFFF", stipple="gray12")

parent.master.bind("<Left>", self.left)

parent.master.bind("<Right>", self.right)

parent.master.bind("<Up>", self.up)

parent.master.bind("<Down>", self.down)

def left(self, event):

x = -720

y = 0

# xx=event.xx

# yy=event.yy

pos = self.canvas.coords('close\_tag')

if pos == [1050.0, 255.0] or pos == [1050.0, 555.0]:

print('left', pos)

self.canvas.move(self.my\_rectangle, x, y)

def right(self, event):

x = 720

y = 0

pos = self.canvas.coords('close\_tag')

if pos == [330.0, 255.0] or pos == [330.0, 555.0]:

print('right', pos)

self.canvas.move(self.my\_rectangle, x, y)

def up(self, event):

x = 0

y = -300

pos = self.canvas.coords('close\_tag')

if pos == [330.0, 555.0] or pos == [1050.0, 555.0]:

print('up', pos)

self.canvas.move(self.my\_rectangle, x, y)

def down(self, event):

x = 0

y = 300

pos = self.canvas.coords('close\_tag')

if pos == [330.0, 255.0] or pos == [1050.0, 255.0]:

print('down', pos)

self.canvas.move(self.my\_rectangle, x, y)

if \_\_name\_\_ == "\_\_main\_\_":

app = SampleApp()

app.mainloop()

**IOT.py**

import json

from itertools import cycle

from tkinter import \*

from math import sin, cos

# from connector import Cortex

from connector import Cortex

import threading

from multiprocessing import Process

class eyeblink(Frame):

def \_\_init\_\_(self, parent, controller):

Frame.\_\_init\_\_(self, parent)

self.packet\_count = 0

self.count = 0

self.id\_sequence = 0

self.switch\_on\_off = None

self.controller = controller

self.c = Canvas(self, width=2085, height=1080, bg="#020A2E")

self.c.pack()

self.create\_good\_rectangle(self.c, 400, 200, 950, 500, 20, 5, '#2C3547')

label = Label(self, text="BLINK", bg='#020A2E', fg='white', font='Arial 50 bold').place(x=80, y=20)

self.light = self.create\_good\_rectangle(self.c, 450, 250, 650, 450, 10, 5, '#5A5D50') # light

self.fan = self.create\_good\_rectangle(self.c, 700, 250, 900, 450, 10, 5, '#5A5D50') # fan

self.c.tag\_bind(self.fan, '<1>', self.fan\_function)

self.c.tag\_bind(self.light, '<1>', self.light\_function)

self.bulb = PhotoImage(file='images/bulb1.png')

self.lighter = self.c.create\_image(500, 265, image=self.bulb, anchor=NW)

self.c.tag\_bind(self.lighter, '<1>', self.light\_function)

self.fann = PhotoImage(file='images/fan1.png')

self.cooler = self.c.create\_image(720, 265, image=self.fann, anchor=NW)

self.c.tag\_bind(self.cooler, '<1>', self.fan\_function)

thread1=threading.Thread(target=self.con)

thread1.daemon=True

thread1.start()

def con(self):

self.fan = PhotoImage(file='images/fan1.png')

url = "wss://localhost:6868"

user = {

"license": "1d10f0b4-65e4-4424-ae8a-a56ae2fa6950",

"client\_id": "uNU5UMKd9eFLYp8JtHr6ZXXaLHg1p6rf1BReVn4N",

"client\_secret": "9wcDtQn2Wubjg7zzlO2tnkx8Hzk1GkLpqZsqgOiuaWsaI0VRkxcxeQ9ZOPrHrNvJ1tlgOAV1XEZ6ooxJ06sRwobXDApRsol08w9YsJWU0fVAieWYp6kHexOlM9OWqXWk",

"debit": 100,

"number\_row\_data": 10

}

self.cortex = Cortex(url,user)

self.headset\_id = self.cortex.query\_headset()

self.cortex.connect\_headset()

self.cortex.request\_access()

auth = self.cortex.authorize()

self.cortex.create\_session(auth, self.headset\_id)

stream\_list=['fac']

self.cortex.subscribe(stream\_list)

while True:

resp = self.cortex.ws.recv()

res = json.loads(resp)

val = res['fac']

eyeact = val[0]

alternator = cycle(("on", "off"))

if eyeact == 'blink':

self.cortex.count += 1

if self.cortex.count == 14:

self.cortex.switch\_on\_off=next(alternator)

self.fan = self.create\_good\_rectangle(self.c, 700, 250, 900, 450, 10, 5, '#FFCE09')

self.fan = PhotoImage(file='images/fan1.png')

self.cooler = self.c.create\_image(720, 265, image=self.fan, anchor=NW)

print(self.cortex.switch\_on\_off)

#time.sleep(0.5)

if self.cortex.count == 24:

self.cortex.switch\_on\_off="off"

self.fan = self.create\_good\_rectangle(self.c, 700, 250, 900, 450, 10, 5, '#5A5D50') # fan

self.fa = PhotoImage(file='images/fan1.png')

self.cooler = self.c.create\_image(720, 265, image=self.fa, anchor=NW)

print(self.switch\_on\_off)

self.cortex.count = 0

#time.sleep(0.5)

self.cortex.packet\_count += 1

def fan\_function(self, event):

self.fan = self.create\_good\_rectangle(self.c, 700, 250, 900, 450, 10, 5, '#FFCE09') # fan

self.fan = PhotoImage(file='images/fan1.png')

self.cooler = self.c.create\_image(720, 265, image=self.fan, anchor=NW)

print("you clicked fan")

def light\_function(self, event):

self.light = self.create\_good\_rectangle(self.c, 450, 250, 650, 450, 10, 5, '#FFCE09') # light

self.bulb = PhotoImage(file='images/bulb1.png')

self.lighter = self.c.create\_image(500, 265, image=self.bulb, anchor=NW)

print("you clicked light")

def create\_good\_rectangle(self, c, x1, y1, x2, y2, feather, res, color): # feather says blunting scale

points = []

# top side

points += [x1 + feather, y1,

x2 - feather, y1]

# top right corner

for i in range(res):

points += [x2 - feather + sin(i / res \* 2) \* feather,

y1 + feather - cos(i / res \* 2) \* feather]

# right side

points += [x2, y1 + feather,

x2, y2 - feather]

# bottom right corner

for i in range(res):

points += [x2 - feather + cos(i / res \* 2) \* feather,

y2 - feather + sin(i / res \* 2) \* feather]

# bottom side

points += [x2 - feather, y2,

x1 + feather, y2]

# bottom left corner

for i in range(res):

points += [x1 + feather - sin(i / res \* 2) \* feather,

y2 - feather + cos(i / res \* 2) \* feather]

# left side

points += [x1, y2 - feather,

x1, y1 + feather]

# top left corner

for i in range(res):

points += [x1 + feather - cos(i / res \* 2) \* feather,

y1 + feather - sin(i / res \* 2) \* feather]

return c.create\_polygon(points, fill=color) # ?

**Email.py**

import imaplib

import email

from email.header import decode\_header

import webbrowser

import os

from tkinter import \*

root =t

# account credentials

username = "bcotapp@gmail.com"

password = "bcotapp9901454"

# create an IMAP4 class with SSL

imap = imaplib.IMAP4\_SSL("imap.gmail.com")

# authenticate

imap.login(username, password)

status, messages = imap.select("INBOX")

# number of top emails to fetch

N = 3

# total number of emails

messages = int(messages[0])

for i in range(messages, messages-N, -1):

# fetch the email message by ID

res, msg = imap.fetch(str(i),"(RFC822)")

for response in msg:

if isinstance(response, tuple):

# parse a bytes email into a message object

msg = email.message\_from\_bytes(response[1])

# decode the email subject

subject = decode\_header(msg["Subject"])[0][0]

if isinstance(subject, bytes):

# if it's a bytes, decode to str

subject = subject.decode()

# email sender

from\_ = msg.get("From")

print("Subject:", subject)

print("From:", from\_)

# if the email message is multipart

if msg.is\_multipart():

# iterate over email parts

for part in msg.walk():

# extract content type of email

content\_type = part.get\_content\_type()

content\_disposition = str(part.get("Content-Disposition"))

try:

# get the email body

body = part.get\_payload(decode=True).decode()

except:

pass

if content\_type == "text/plain" and "attachment" not in content\_disposition:

# print text/plain emails and skip attachments

print("===============================================")

print(body)

imap.close()

imap.logout()

**5.4 Observation and Result**

Some Example:-

Facial Expression:-

This feature has training frame and live frame that more you train more accuracy obtained. This example demonstrates how an application can use the Facial Expressions detection suite to control an animated head model called BlueAvatar. The model emulates the facial expressions made by the user wearing an Emotiv headset. As in Example 1, Facial Expressions Demo connects to Emotiv EmoEngine and retrieves EmoStates for all attached users. The EmoState is examined to determine which facial expression best matches the user’s face. Facial Expressions Demo communicates the detected expressions to the separate BlueAvatar application by sending a UDP packet which follows a simple, pre-defined protocol. The Facial Expressions state from the EmoEngine can be separated into three groups of mutually-exclusive facial expressions:

1. Upper face actions: Surprise, Frown
2. Eye related actions: Blink, Wink left, Wink right
3. Lower face actions: Smile, Clench, Laugh

Here the training frame has 5 facial commands that needs to be trained, if the live feed is not

accurate then train neutral more. The live feed takes 3 sec to initiate connection

We can get result as like shown in below figure :-

A picture containing monitor, screen, television, computer

Description generated with very high confidence

**Figure 5. Facial Expressions**

**CONCLUSION AND FUTURE WORK**

**CHAPTER 6**

**CONCLUSION AND FUTURE WORK**

**6.1 Conclusion**

In this project, We have come up with the idea of getting signals from brain to do the external works. As we know that BCOT is a collaboration between a brain and machine learning that enables signals from the brain to direct some external activity, such as control of a computer or a mobile or any other devices. which fused single-trial EEGs from multiple subjects to improve the overall BCOT system performance. By comparing system designs and data analysis methods, this study showed that a distributed paradigm combined with a Voting classifier is a practical solution for implementing a collaborative BCOT system. The feasibility and efficacy of the proposed BCI system was demonstrated through a collaborative BCI that could accelerate motor decision-making of a cue-guided reaching movement. The classification accuracy of the system showed a significant improvement over that of the single-subject BCI. Furthermore, the collaborative BCI allowes the subject's reaching direction to be estimated much earlier than his/her actual motor response. In summary, this study designed and demonstrated the use of the collaborative BCI technology to improve BCOT and human performance in natural environments.

**6.2 Future Work**

It is quite probable that in the future most of our appliances will be controlled directly through our wishes or the brain and this project stands as an affirmation to that vision. Signals from the brain can be further studied and the technology can be refined to bring about more specific results. The scope of the project was primarily to establish control through no physical motion on part of the user and it has been successful in doing so but it has also laid a foundation for many applications which would greatly improve the standard of life for all.

**REFERENCES**

[1] Gerwin Schalk and Eric C. Leuthardt, “Brain-Computer Interfaces Using Electrocorticographic Signals,” IEEE Reviews In Biomedical Engineering, vol. 4, October 2011.

[3] Luzheng Bi, Xin-An Fan, and Yili Liu, EEG-Based BrainControlled Mobile Robots: A Survey, IEEE Transactions On Human-Machine Systems, vol. 43, no. 2 , March 2013.

[3] J. Yue, J. Jiang, Z. Zhou & D. Hu, SMR-Speller: A novel Brain-Computer Interface spell paradigm. In Anonymous Computer Research and Development (ICCRD), 2011 3rd International Conference on. (). , 2011, 187-190.

[4] L. R. Hochberg, M. D. Serruya, G. M. Friehs, J. A. Mukand, M. Saleh, A. H. Caplan, Almut Branner, David Chen, Richard D. Penn, John P. Donoghue, “ Neuronal ensemble control of prosthetic devices by a human with tetraplegia,” Nature, Vol. 442(13), pp. 164-171, 2006.

[5] <https://openbci.com/forum/>

[6] <https://sccn.ucsd.edu/>

[7] <https://emotiv.gitbook.io/cortex-api/advanced-bci>

**APPENDIX**

**APPENDIX A**

**SNAPSHOTS**

**A picture containing appliance, indoor, sitting, small

Description generated with very high confidence**

**Figure A.1. Emotiv insight device**

**A close up of a sign

Description generated with very high confidence**

**Figure A.2. Home page**

**A person taking a selfie

Description generated with high confidence**

**Figure A.3. Output of Blink for ON/OFF**

**A screenshot of a cell phone

Description generated with very high confidence**

**Figure A.4. Features Page**

**A screen shot of a person

Description generated with very high confidence**

**Figure A.5. Output for Facial Expressions**

**A screenshot of a boy

Description generated with high confidence**

**Figure A.6. Output for Act To Speak**