Flinders
UNIVERSITY
ADELAIDE · SOUTH AUSTRALIA

Brain activity visualisation using a physical 3D Brain with LEDs

By

Mileeni Chowdary Chandra

Thesis
Submitted to Flinders University
for the degree of

Biomedical Engineering

College of Science and Engineering
Date: 23-05-2024

TABLE OF CONTENTS

EXECUTIVE SUMMARY

Visualising and analysing the functioning of the brain during different tasks and activities is a fascinating field of study. Previous studies of brain activity visualisation were mostly done on a 2D computer screen where there are different tools present for 3D and 2D analysis of brain data. The 2D screen gives less spatial information as all the data cannot be seen at once. The physical 3D model developed in the research helps in providing a better overall view of brain activity which is easy to eyes and utilises pre-recorded EEG data. This physical model has an LED strip placed inside it which receives the converted EEG to LED data.

The DotStar LED strip and Mega328p microcontroller used to drive the strip are carefully selected and tested for their functionality. A program to convert the EEG data to LED data is developed using MATLAB and Arduino IDE and the data is transmitted to an LED strip through a wired connection. A 10-10 EEG graph with 74 electrode placement locations is utilised for the placement of the LEDs and provides good spatial information. The time taken for transmission of converted EEG data to the LEDs was optimised to be 92.5ms due to the restrictions involving hardware components. The model developed shows good visualisation of EEG data at correct anatomic locations. This research can be further used for visualising live brain activity and can also be used for comparison of healthy and unhealthy brains.

DECLARATION

I certify that this thesis:

1. does not incorporate without acknowledgment any material previously submitted for a degree or diploma in any university.
2. and the research within will not be submitted for any other future degree or diploma without the permission of Flinders University; and
3. to the best of my knowledge and belief, does not contain any material previously published or written by another person except where due reference is made in the text.

Signature of student: [redacted]

Print name of student........Mileeni Chowdary Chandra

Date.....22-05-2024

I certify that I have read this thesis. In my opinion, it is/is not (please circle) fully adequate, in scope and in quality, as a thesis for the degree of Biomedical Engineering. Furthermore, I confirm that I have provided feedback on this thesis and the student has implemented it minimally/partially/fully (please circle).

Signature of Principal Supervisor.......[illegible]...

Print name of Principal Supervisor......Kenneth Pope..

Date…………..23/05/2024....................................

ACKNOWLEDGEMENTS

I would like to express my deepest appreciation to my supervisor, Associate Professor Kenneth Pope, for bringing his knowledge and experience to this thesis. I thank him for having good faith and patience in me for completing this project.

I am also thankful to the Engineering services team who guided me in completing the prototype of this project.

I would also like to thank my parents and my sister for providing me with support and understanding during the last few months.

Lastly, I would like to acknowledge Flinders University for allowing me to study here and enrich myself with knowledge.

LIST OF FIGURES

INTRODUCTION

The human brain is a complex network of neurons that contain electrical signals. It is an adaptive system where coordinated neural activity takes place to generate various behaviours and actions (Shine *et al.*, 2019). It is fascinating to know how different tasks and activities generate different neurological patterns in the brain. All the brain activity is processed in milliseconds. This data can be captured using different techniques like electroencephalography (EEG) (Michel and Murray, 2012), and magnetoencephalography (MEG) (Brookes, Woolrich and Barnes, 2012) in different resolutions. However, in this research EEG technique is used due to its inexpensiveness and also the availability of prerecorded EEG data in the database.

The brain activity can be visualised using different 2D and 3D techniques. However, 3D modelling helps in the better understanding of spatial and temporal characteristics of the EEG data. Research showed that 3D visualisations have already been implemented using various software tools but only on a 2D screen (Mammone *et al.*, 2010; (Eck and Goebel, R., Esposito, F. Formisano, E., 2006) or using Virtual Reality (Pester *et al.*, 2022) . This thesis demonstrates the visualisation of prerecorded EEG data on a physical 3D model with LEDs placed inside it, as this model provides more spatial information and is easy on the eyes. By controlling LEDs inside the 3D brain, we can present brain activity in real-time by mapping the EEG data samples to the LEDs. There was previous research done with LEDs in the brain but they provide low spatial resolution which is not the case for this thesis (Sanmati Chaudhary, 2018; linda grefen, no date). This project is a continuation of previous students' work, where the 3D model of the brain was constructed, and the control of an LED strip was partially achieved. The previous work mainly focussed on researching the different hardware components required to build the brain and making decisions on the software to be used. The components required were acquired and the LED strip was partially tested for its functionality without using any EEG data.

In this thesis, prerecorded EEG data is loaded and processed to give LED colour data using the signal processing software tool MATLAB. The data is then sent using a wired connection to the physical 3D brain using a serial connection between the system where MATLAB software and the Arduino IDE software is present. The performance of the hardware components is analysed, and the visualisation of EEG data is evaluated at different transmission speeds. Multiple steps were taken throughout the research for this visualisation. The methodology followed is (1) assembly of hardware components, (2) testing of the components, (3) testing of the EEG to LED data processing code, (4) testing of different data transmission speeds between the system and the LEDs, and (5) the placement of the LEDs inside the model and testing the functionality. These steps resulted in the establishment of 3D visualisation of prerecorded EEG data.

LITERATURE REVIEW

To date, there are many techniques which are being used for presenting EEG data in a readable format. The formats can be using graphs, sliders, light (LEDs), and so on (Liu, Dabbish and Kaufman, 2017). They are usually presented primarily using a 2D format and a 3D format (Badea, Kostopoulos and Ioannides, 2003; Birgani and Ashtiyani, 2007; Anderson *et al.*, 2018). Different approaches are being used to display these formats on a computer screen.

Tools and algorithms like Convolution Neural Networks (CNN), which is used for image generation by processing EEG data, and Functional Brain Network Analysis and Visualisation (FBNAV), which is used to visualise various data analyses, are being used in both 2D and 3D visualisation (Vijayalakshmi *et al.*, 2014; Xiaoyan *et al.*, 2018). When using CNNs for seizure detection from multi-channel EEG data in 2D and 3D images, 3D CNNs provided an accuracy of 92.37%, which exceeded the accuracy of 89.9% from 2D CNNs (Xiaoyan *et al.*, 2018). It was also seen that 3D CNNs provided more multi-channel information and showcased a 3D format provides more scope to visualise the vast amount of EEG data and provides more accurate classification (Xiaoyan *et al.*, 2018). CNNs also reduce the amount of time consumed to generate 3D images using hand-engineered methods as the model is usually trained. CNNs require a lot of time (high computational cost) and a lot of data to train well. While a pre-trained CNN could be used, the similarity of the data it was trained on and the data used subsequently may not be sufficient for useful results (Sadiq *et al.*, 2022).

Another technique found was the use of the Short-Term Maximum Lyapunov Exponent (STLmax) parameter, which clusters the electrode power data according to different frequencies to generate images (Mammone *et al.*, 2010). This technique can only be used to generate 2D images. However, various tools display data in 3D formats, such as Brain Voyager (Eck and Goebel, R., Esposito, F. Formisano, E., 2006) and CURRY (*Curry 8 | NEUROSPEC AG Research Neurosciences*, no date). Additionally, Trans3D (Blinowski, Kamiński and Wawer, 2014) is a free tool that facilitates 3D visualisation of brain activity transmission. 3D visualisation is more attractive as it attracts higher attention and uses more working memory (Malik *et al.*, 2015). However, all of these visualisation techniques are limited to a 2D computer screen.

Then comes the Virtual Reality (VR) approach where a person can interact with the brain data and explore it using handheld controllers (Pester *et al.*, 2022). It proved that people found it more useful to interact and understand their data than watching it in 3D on a 2D screen. This approach can be expensive, and only one person can visualise the data this way at a time, excluding group discussion of data.

Hence for this project, it was decided to use a physical 3D model with LEDs which is a cheaper alternative and provides a similar visual experience. This also ensures there can be multiple viewers and hence more interaction, leading to a better understanding of the data. There is no prior published research studying the use of LEDs for the 3D visualisation of brain activity. This project is limited to the use of prerecorded EEG data, i.e., not the display of data as it is being sampled live.

METHODOLOGY

The research was carried out in different steps for building the 3D model, establishing a connection between system and data source along with the testing of the prototype in different stages.

Testing of hardware components

For the building of the prototype, different electronic components such as LED strip, and microcontrollers were required. For the physical brain, a 3D model was used which was printed using the 3D printing machine available at the university. This 3D-printed brain is a replica of the human brain and has already been designed and made available in previous research. (Figure 1) represents the 3D model used for the placement of LEDs.

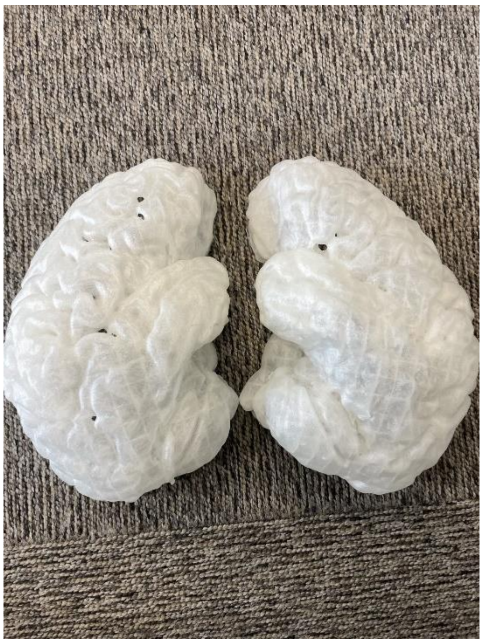


Figure 1: 3D printed brain.

The LEDs that needed to be placed inside the 3D brain were selected to be an RGB DotStar LED strip which has 144 LEDs and was 0.5m in length. Each LED approximately represents an EEG electrode used for collecting the brain data and 144 electrodes generally do a good job at covering all electrodes. The EEG technique generally has a maximum of 128 electrode positions on the scalp and they are optimal for understanding the brain activity. If there are more number of LEDs, it becomes difficult to place them inside the 3D brain and also hinders the visualisation of data. DotStars provides a fast data transfer rate of 8MHZ on Arduino along with fast PWM rates. They are easy to interface with many devices and need only two pins for controlling the data (*Adafruit DotStar LEDs*, no date). The LEDs are usually available individually, in a strip, and as a grid. The DotStar strip was selected as it can be easily cut into any length and is flexible to be placed in the 3D brain. As the length of the strip is long and contains a huge number of LEDs, they consume power. The 144 LED strip has an approximate peak power of 43 watts which is 5V 8.6A. Most microcontrollers available cannot power the LED strip as they have a low output current of approximately 40mA. So, a power adapter of 5V 8. A has been connected to one end of the strip. (Figure 2) shows the LED strip which has two control pins namely DATA pin (D0) and CLOCK pin (C0) along with power pins 5V and GND (*Adafruit DotStar LEDs*, no date).

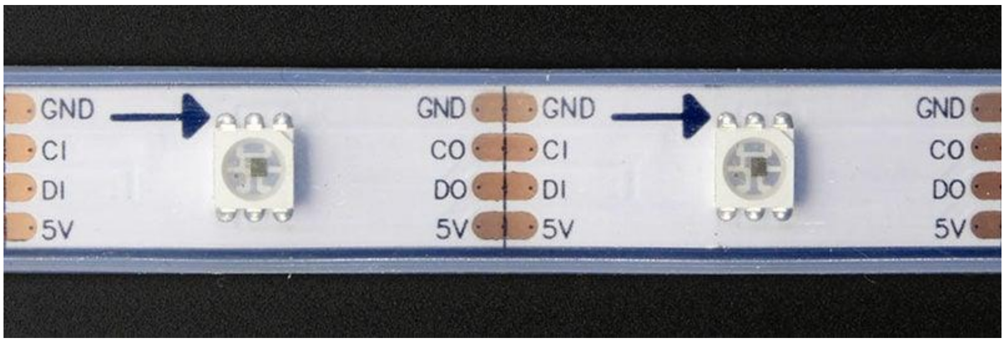


Figure 2: DotStar LED strip containing 144 LEDs.

Lastly, a microcontroller unit is used to transfer the EEG data in a readable format to an LED strip. Different microcontrollers were used to initially test the LED strip. They are Funduino UNO, ESP32, and ATMega328p. Funduino UNO and Mega328p operate on a voltage of 5V whereas Esp32 operates on 3.3V. As the LED strip runs off 5 V, the Esp32 and other 3.3 V microcontrollers were not further considered. Both the Funduino UNO and Mega328p have an SRAM of 2KB while ESP32 has an SRAM of 520KB (*Arduino vs ESP8266 vs ESP32 Microcontroller Comparison*, 2019).

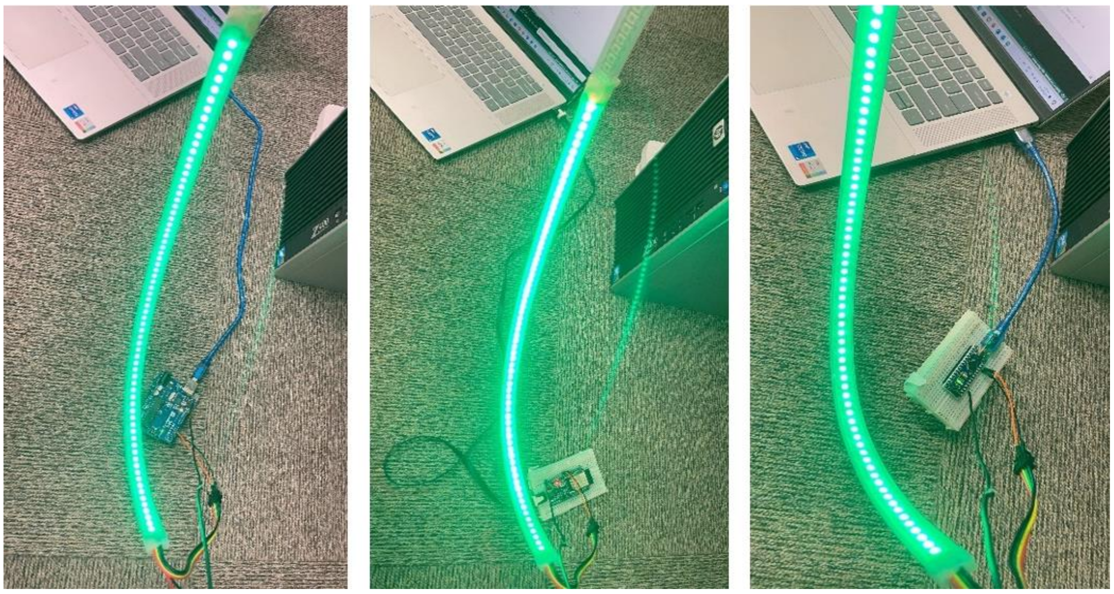
Testing of Microcontrollers with LED strip

Each of the above microcontrollers was then tested using the LED strip. As the microcontrollers need to be programmed for this purpose, Arduino IDE software v2.3.3 was installed in the system along with the necessary CP210X drivers for the USB connections. The syntax used in the software is like C++ and is very simple to read and write (*Arduino Software (IDE)* | *Arduino*, no date).

A simple program was written for a DotStar LED strip by including the required Adafruit DotStar library. A strip.fill() function was used to send red, blue, and green colours simultaneously to the whole strip of 144 LEDs and a baud rate of 115200bps was used.

Firstly, this program was tested using Funduino UNO where the DATA pin and CLOCK pin of the LED strip were connected to the Digital pins 3 and 4 of UNO respectively. The necessary power connections were also given. Later, the program is uploaded to the microcontroller board using a USB wire through a serial communication port. It was seen that all the LEDs were not lighting up and only 100 LEDs were in working condition as seen in (Figure 3a).

Next, the program was tested using an ESP32 board. The same program was used with a change in pin connections. The necessary board for ESP32 was installed in the IDE and the program was uploaded to the controller board. It was observed that even ESP32 was not able to light up all 144 LEDs as shown in the (Figure 3b). Finally, the program was tested using a Mega328p board with a change in the notation of the pin connections. The necessary Atmel AVR boards were installed to the IDE and the program was uploaded via USB connection. It was observed that all 144 LEDs in the strip were functioning, and the colour change from red to blue to green was seen as shown in (Figure 3c).



a *b* *c*

Figure 3 a) Funduino UNO module with LED strip, b) Esp32 module with LED strip, c) Esp328p module with LED strip.

The Esp32 controller was not able to propagate data to all the LEDs despite having higher speed and SRAM was because of the differences in operating voltages. The Esp32 microcontroller operated at 3.3V whereas the LED strip operated at 5V. Both the components were not compatible with each other, hence the failure. The Funduino microcontroller has the same operating voltage as the LED strip but was not able to successfully propagate data. This can be due to it being a clone of the actual Arduino UNO. So, the Mega328p microcontroller was selected as the hardware programming unit for further development of the prototype. A wireless connection was not done in this research as the chosen hardware does not have an inbuilt Wi-Fi connection. The Esp32 controller was seen as a potential for wireless connection initially due to the presence of inbuilt Wi-Fi, but it was observed that it was not compatible with the LED strip. The use of two microcontrollers was seen as too expensive.

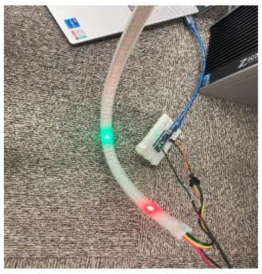
Functional testing of LED strip

After the Mega328p microcontroller was finalised for prototype, the next stage of testing involved the DotStar LED strip. The functionality of the LED strip was tested by writing simple programs in IDE. Three types of programs were written for this purpose.

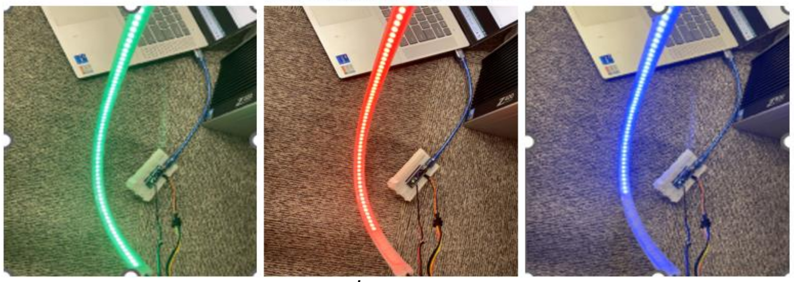
Firstly, the individual addressability function of the LEDs was tested by using a strip.setPixelColor() function which lights up the particular addressed LED with a given intensity from 0-255. LEDs 5 and 25 were set to red and green respectively and the program was uploaded to the microcontroller. (Figure 4a) showed that the LEDs in this strip can be individually addressed and can be set to different colours and intensities.

Next, the colour-changing property was observed for the whole strip as done in the previous testing using the strip.fill() function with varied intensities from 0-255. The change was successfully observed.

Finally, the strip was tested to check if blocks or sets of LEDs could be addressed. The program written was such that the LEDs lit up from different start points using different colours for each start point for the rest of the strip. (Figure 4b) shows the change of colours from green to red to blue with LEDs starting from 0, 10, and 20 respectively.



a



b

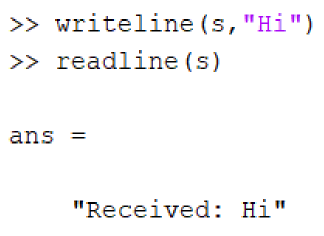
Figure 4: a) Individual addressing of the LED strip, b) The three colour sequences of the LED strip from green to red to blue.

By doing the above testing, the working and different functionalities of using a DotStar LED strip were explored.

Serial Communication between MATLAB and Arduino

The prerecorded EEG data which is needed for processing is stored in a database in the local system called the EEG3 Toolbox. This signal processing toolbox can be accessed using MATLAB using predefined functions for loading the data. The required EEG data can only be accessed using the system where the EEG3 toolbox is present, but it also provides scope for performing complex analysis and calculations of the data which gives it an advantage. As MATLAB is required, the existing MATLAB R2019b version has been used for accessing the data. A path was established to the EEG data present in the R drive of the system for easy accessibility. The EEG data needs to be sent to Arduino to establish a connection to the LED strip. For this purpose, serial communication between MATLAB and Arduino was needed.

A common serial communication port was given to both MATLAB and Arduino through which they can communicate. The baud rate was initially set to 115200bps for both Arduino and MATLAB. It is necessary to have the same baud rate to ensure data transmission. Next, text messages were sent between MATLAB and Arduino to see if they were being received. (Figure 5) shows the program written in Arduino and MATLAB where when a “hi” was sent as a text to Arduino and the controller detects the message, it sends back "Received" as a response. It was seen that character-type data could be sent over the serial port.



```
>> writeline(s,"Hi")
>> readline(s)

ans =

    "Received: Hi"
```

Figure 5: Message received in the command window of MATLAB during serial communication between MATLAB and Arduino

Later, an array of integers was sent to Arduino from MATLAB and vice-versa to see if the data type integer could be transmitted. It was observed that the data could be sent using uint8 datatype notation. The testing ensured the transmission of integer data.
The final testing was done using an LED strip, a simple program was written in MATLAB to change the colour of the whole strip from red to blue to green. The data sent was in the form

of bytes. It was observed that the strip was displaying some part of the previous colour after the colour change. It was due to the high speed of data transmission and no added delays when opening the serial connection. To test this, a simple intensity change test was conducted to change the intensity of 50 LEDs. The red colour of the LED strip was changed to 5 different intensities starting from 50 to 250 in steps of 50. Each LED was sent 3 bytes of intensity data which means that 150 bytes of information was sent for 50 LEDs. This data was verified by sending a return message of the number of bytes received by the Arduino to MATLAB.

An initial delay of 1 sec was introduced when the serial communication port was opened for Arduino and MATLAB. This delay was introduced to give enough time for the Arduino to receive the data sent from MATLAB. It is because the serial port connected is being switched between both MATLAB and Arduino when the data is sent and received. Each time the serial port connection is established in Arduino, the microcontroller is refreshed which consumes some time. During this time, the data sent from MATLAB can be lost. By adding delay, enough time is given for Arduino to reset and start receiving the data.

This program is now tested using different baud rates to check the speed at which the data can be transmitted without incurring any loss. It was seen that at baud rates 115200bps, 57600bps, 19200bps, and 9600bps the whole data was not being received. If the speed is set lower than that the data transmission would be slow. An initial delay of 2 seconds was then used, and it was seen that at speed 19200bps the 150 bytes of data is being received by MATLAB.

By this testing, it was observed that as the use of LEDs increases, there is a requirement to adjust the speed of transmission according to it.

Mapping of EEG data to LED data

As the serial communication between MATLAB and Arduino is established, the next step is to send the EEG data to an LED strip in the form recognised by the LEDs. The EEG data was converted to LED data in MATLAB and then sent over to Arduino. The processing of data was done in MATLAB as it provides various functions and scope for signal processing and data processing. It is also easy to retrieve the EEG data using the EEG3 toolbox.

Initially, the startup program was run to initiate and establish a path to the EEG3 toolkit. Different preferences and paths were set to the required files in the toolbox so that the data was always available for access.

For this test, a data sample of finger tapping data of participant '0113' which had 127 channels of information was used which can be seen in (Figure 6). All the EEG data stored in the toolbox has been ethically sourced by the researchers and students by using the EEG technique. A serial communication port object was created in MATLAB with a data transmission speed of 19200bps and an initial delay of 2 seconds. The data is then loaded into MATLAB using a predefined function eeglocal.gamma.loadSubject(participant, task) where the participant is ‘0113’ and the task is finger tapping done by 0113. This function loads the EEG data present in the gamma path stored in the R drive. After loading the data, a start and stop time was established for the data using a timebase function for the 127 channels used.

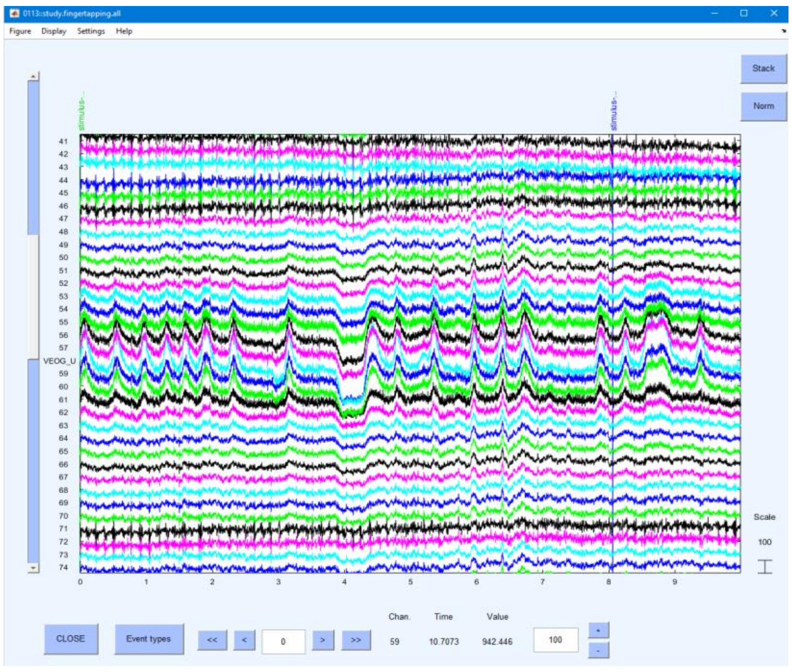


Figure 6: Sample EEG data containing the study of finger tapping of a participant used for testing seen in MATLAB

A while() loop was established to slide one block at a time for the procured start time in the steps of 0.1s. For each loop, all the 127-channel data was processed until the end of the procured stop time. The start and stop time where there was a stimuli observed in the participant doing the task can be seen in (Figure 7). The colours for all the LEDs were established by using the RMS function on the data. This RMS function converts the matrix of EEG data to a row vector with an RMS value for each column. This RMS data was then multiplied to a row of intensity data [255 255 255]. This converts the EEG data into different intensities with each intensity containing 3 bytes of data. This intensity data was stored in a function named colours and sent as LED data in the form of a 1-by-n array. The serial port object was deleted after the end of the loop disconnecting the control of MATLAB to Arduino after data was sent.

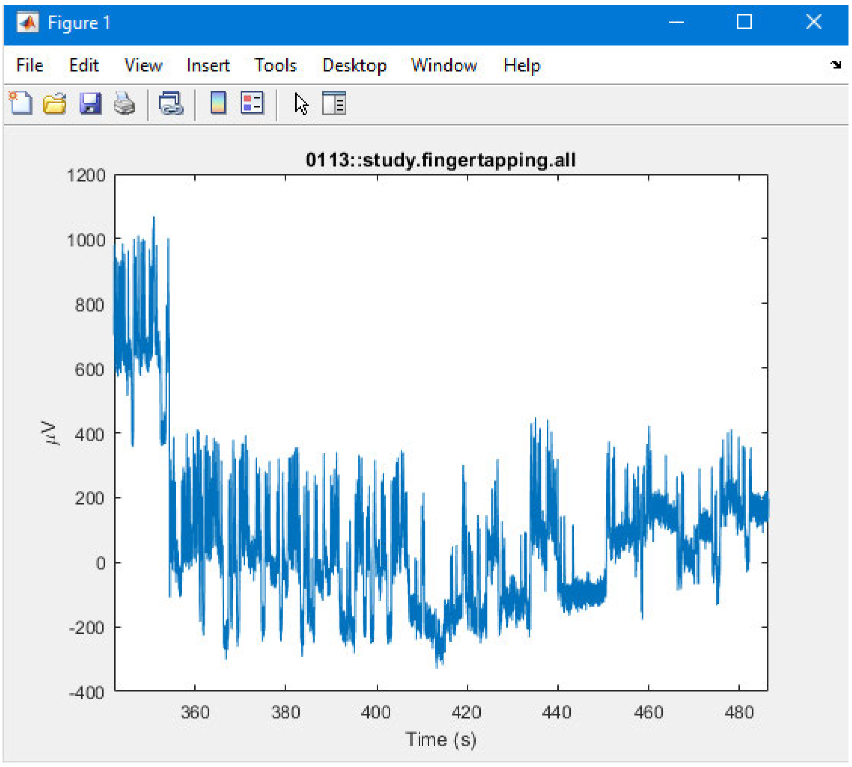


Figure 7: Sample EEG data stimuli observed in a particular time interval used for processing LED data.

In Arduino IDE, an initial delay was introduced after the opening of serial port communication which is based on the number of LEDs used and the baud rate. The data being sent from MATLAB was stored temporarily in an array called buff by using the function Serial.readBytes(). The received buffer data was transmitted to the LEDs using a strip.setPixelColor() function in a continuous loop until all the data is sent to the strip. The strip.setPixelColor function takes the intensities sent from MATLAB and uses a colour map according to the data received.

Finally, the program was uploaded to Arduino Mega328p over a USB cable which was connected to the serial port used by both MATLAB and Arduino. Then, the program in MATLAB is run to establish a connection with Arduino. The LEDs changed colour and this data was compared to the image of the Colours function in MATLAB. Some data was missing between blocks of data. The testing protocol of serial communication was used here again by sending back 127x3 bytes of data to MATLAB from Arduino. The number of LEDs used was slowly reduced and it was seen that at 19200bps the information was sent to only 80 LEDs as all the bytes were received. Beyond 80 LEDs, it was seen that some data was being missed by Arduino. This is mostly likely due to buffer overflow causing data to be lost in the arduino. The data transmission speed was then reduced to 9600bps, and the bytes received were observed. All the 127x3 bytes of data were received at this speed. The speed of 9600bps was tested for the LED colour program and it was seen that the image of the Colours in MATLAB was similar to the LED strip colour change.

LEDs in the 3D model of the brain

The remaining stage of prototyping is the placement of LEDs in the physical 3D model of the brain. The LEDs are placed in the model according to the 2D EEG graph which is used as a reference for EEG electrodes to collect brain data. There are different standards used for electrode placement. The international standard system generally used clinically is a 10-20 system which uses 21 electrodes with 20% of the total skull distance between each electrode and 10% of distance between the outer electrodes and the nasion and inion anatomical landmarks. However, it does not provide high density information (TMSI, 2022). A 10-5 standard system for electrode placement on the scalp was used as a reference which can be seen in (Figure 8a). It has 145 electrodes and uses 5% distancing between electrodes (‘High-density EEG electrode placement | Robert Oostenveld’s blog’, 2006). The EEG data available for testing is collected using this system and provides ample information.

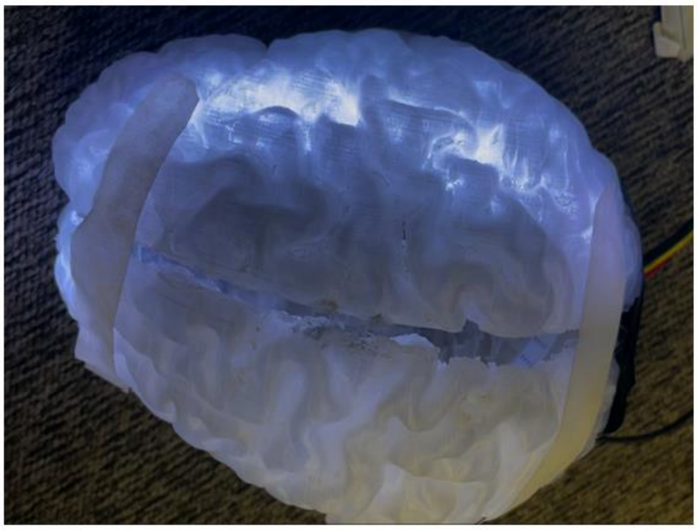
Removed due to copyright

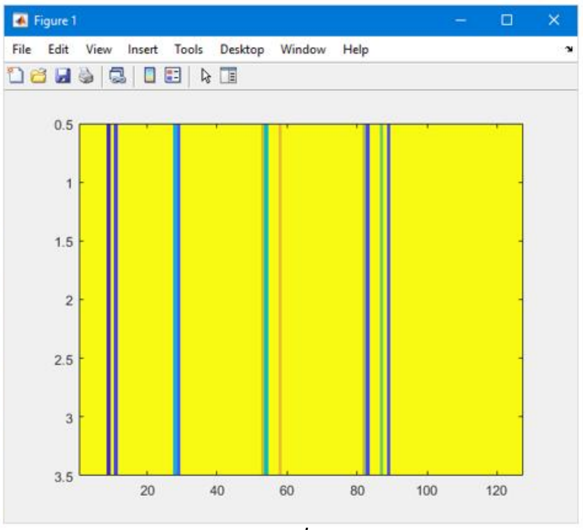
Figure 8: a) A 10-5 standard 2D EEG graph for electrode placement, and b) A 10-10 standard 2D EEG graph for electrode placement. (Jurcak, Tsuzuki and Dan, 2007)

This 10-5 system can be increased to a 10-10 system as seen in (Figure 8b) (Jurcak, Tsuzuki and Dan, 2007). This 10-10 system provides information on only 74 electrodes compared to the 145 electrodes in the 10-5 system. However, it still provides high resolution of information and does not provide overlapped information (Jurcak, Tsuzuki and Dan, 2007). For the placement of LEDs, it is beneficial to use the 10-10 system as it is much easier to place them inside the physical brain without cluttering them. Based on this, it was decided to use the 10-10 system for LED placement. The electrode positions were marked and calculated on the physical 3D brain. The sleeve on the LED strip was removed and the strip was cut into strips and connecting wires were used to attach one strip to another. The strips were then glued to the physical brain. The LED program is modified to include only the given number of 74 electrodes and the speed is increased to 19200bps as the LEDs are less than 80.

RESULTS

Figure 9a gives the visualisation of brain activity using LEDs inside the 3D brain. Figure 9a shows the EEG data for a 10-10 system where 74 electrode placements are used. The LED intensity in the 3D brain varies according to the image produced on the computer using MATLAB which can be seen in (Figure 9b). The baud rate is increased to 19200bps as the LEDs are less than 80.

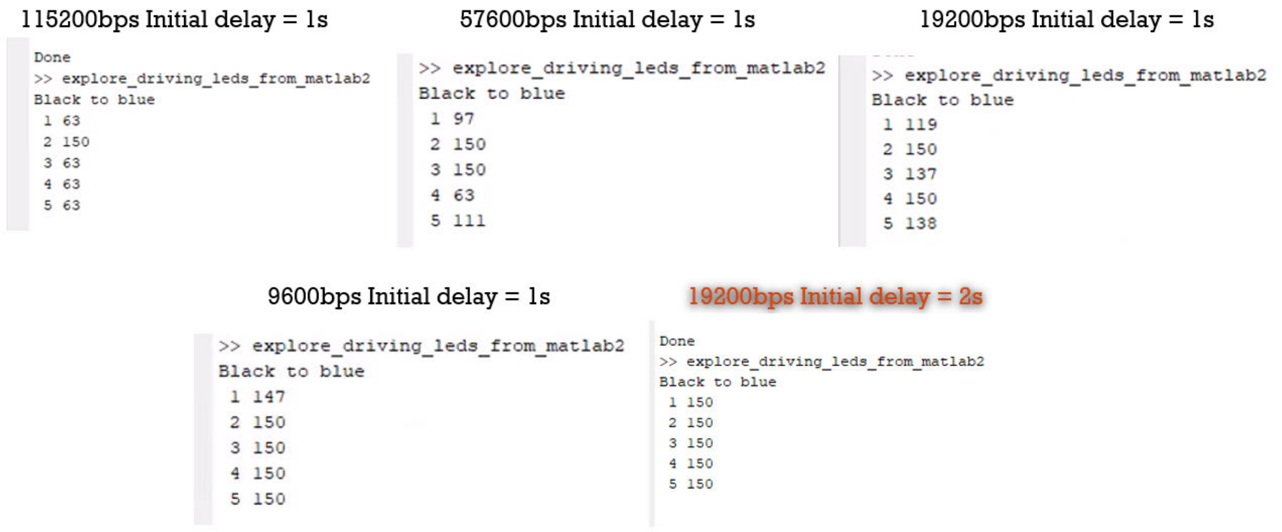

a



b

Figure 9: a) Visualising 3D printed brain with LEDs inside it, b) the image of LED data in MATLAB.

The LED data is sent to the physical brain from the system using a micro-USB connection. This shows that a wired connection has been established between them for data transmission. Figure 10 shows the different transmission speeds of LED data from MATLAB to Arduino when the intensities of the strip were varied for 50 LEDs from 50 to 250 in steps of 50 and the amount of data received by Arduino was observed.



115200bps Initial delay = 1s

```
Done
>> explore_driving_leds_from_matlab2
Black to blue
 1 63
 2 150
 3 63
 4 63
 5 63
```

57600bps Initial delay = 1s

```
>> explore_driving_leds_from_matlab2
Black to blue
 1 97
 2 150
 3 150
 4 63
 5 111
```

19200bps Initial delay = 1s

```
>> explore_driving_leds_from_matlab2
Black to blue
 1 119
 2 150
 3 137
 4 150
 5 138
```

9600bps Initial delay = 1s

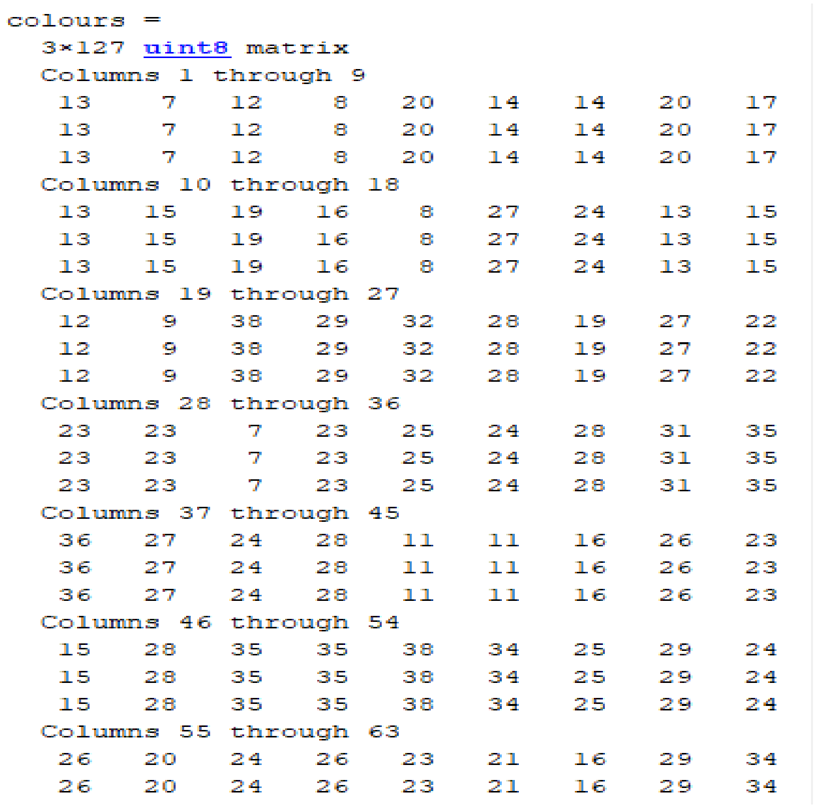
```
>> explore_driving_leds_from_matlab2
Black to blue
 1 147
 2 150
 3 150
 4 150
 5 150
```

19200bps Initial delay = 2s

```
Done
>> explore_driving_leds_from_matlab2
Black to blue
 1 150
 2 150
 3 150
 4 150
 5 150
```

Figure 10: Total bytes of data received by LEDs for different Baud rates and delays in the command window of MATLAB.

The LED intensity is varied according to the respective EEG data which can be seen in (Figure 11). The figure shows LED data information of a block of data at an instance of time. The intensities vary from 0 to 255 for 127 electrodes. These intensities are sent to Arduino via serial communication to change the colour of LEDs.



```
colours =
  3×127 uint8 matrix
  Columns 1 through 9
    13     7    12     8    20    14    14    20    17
    13     7    12     8    20    14    14    20    17
    13     7    12     8    20    14    14    20    17
  Columns 10 through 18
    13    15    19    16     8    27    24    13    15
    13    15    19    16     8    27    24    13    15
    13    15    19    16     8    27    24    13    15
  Columns 19 through 27
    12     9    38    29    32    28    19    27    22
    12     9    38    29    32    28    19    27    22
    12     9    38    29    32    28    19    27    22
  Columns 28 through 36
    23    23     7    23    25    24    28    31    35
    23    23     7    23    25    24    28    31    35
    23    23     7    23    25    24    28    31    35
  Columns 37 through 45
    36    27    24    28    11    11    16    26    23
    36    27    24    28    11    11    16    26    23
    36    27    24    28    11    11    16    26    23
  Columns 46 through 54
    15    28    35    35    38    34    25    29    24
    15    28    35    35    38    34    25    29    24
    15    28    35    35    38    34    25    29    24
  Columns 55 through 63
    26    20    24    26    23    21    16    29    34
    26    20    24    26    23    21    16    29    34
```

Figure 11: EEG to LED converted data.

The data transmission speed used in the program is 19200bps. This is the baud rate which was established after repeated testing to observe if all the bytes of information are being received by the LED strip. The speed of transmission to send 127LEDS x 3bytes = 381 bytes of data can be calculated using the formula:

$$Time = \frac{Number\ of\ bits}{Baud\ rate}$$

Each byte is equal to 8 bits of data and the time taken to send 381 bytes at 9600bps is:

$$Time = \frac{381 * 8bits}{9600bps} = \frac{3200\ bits}{9600bps} = 0.333\,\mathrm{sec} = 333.34\ milliseconds$$

The established data transmission speed between MATLAB and Arduino is 166.7ms.
For 74 LEDs, the baud rate is changed to 19200bps the speed of transmission is:

$$Time = \frac{74 * 3 * 8bits}{19200bps} = \frac{1776\ bits}{19200bps} = 0.0925\,\mathrm{sec} = 92.5\ milliseconds$$

DISCUSSION

The visualisation of the EEG data on a physical 3D model has been established with the use of LEDs. The data was sent from the system to the model using a wired connection which is the micro-USB connection. A single brain activity of finger-tapping function was used to see the working of the 3D model and it was seen that the wired connection was successful.

The intensities of LEDs were derived by converting the EEG data to LED data in MATLAB by using the RMS of the EEG data and multiplying it with the peak intensity of the LED which is 255. This LED data is transmitted over the USB to the LED strip at a baud rate of 9600bps. To send 127x3= 381 bytes of data it would take approximately 333.3 milliseconds. This transmission speed is more than 50ms which was initially the aim of this project.

During 3D model assembly, only 74 LEDs were used for ease of placement, so the time taken for transmission is further reduced to 92.5 milliseconds. The speed was optimized as much as possible for the hardware used. The Mega328p has only 2Kb SRAM and 16Hz clock speed which is not fast enough to send huge amounts of data.

The placement of the LED strip inside the 3D brain was done according to 10-10 electrode placement. Though the LED is placed as accurately as possible near the correct anatomical location, there is still some scope for error. All the positions of the 10-5 electrode cap were not utilised for the LED placement. The 10-10 electrode placement provides sufficient high-resolution information required and was thought to be enough for this project.

The wireless connection can be established in the future by using a Mega2560 microcontroller with an ESP8266 Wi-Fi module which has 8KB of storage space and high amount of speed. Or a 5V capability microcontroller can be researched which has compatibility with LED strip. The speed of the data transmission can also be increased by replacing it with a microcontroller with high speed and storage capability.

CONCLUSIONS

A wired communication between the 3D physical model of the brain and the system where EEG data has been stored is achieved through this research. The hardware components used in this study were analysed and tested successfully. The prerecorded EEG data is converted to LED data using MATLAB and has been sent to its addressable location on the 3D model using Arduino IDE software. The speed of transmission of the EEG to LED colour data from the system to the model has been optimized as much as possible. This could not be further optimised due to the restrictions of hardware components (USB, microcontroller, software) used. The visualisation of EEG data at an accurate anatomic location on the 3D model has been achieved with the further scope of increasing resolution.

FUTURE WORK

This research work can be used for the inclusion of live EEG data as the data source. The EEG data can be collected and processed live from a person when performing a given task. The person can then visualise their brain activity to this performed task in real-time.

Another study is the use of this prototype for the analysis of brain activity of a person who is unhealthy and suffers from brain disorders. The EEG data collected from the unhealthy person when performing a certain task can be compared with a healthy person performing the same task using 3D visualisation. This helps in getting a clear understanding of brain activity which can be used for clinical studies for tracking abnormal patterns and for education purposes.

BIBLIOGRAPHY

- *Adafruit DotStar LEDs* (no date) *Adafruit Learning System*. Available at: https://learn.adafruit.com/adafruit-dotstar-leds/overview (Accessed: 20 May 2024).

- Anderson, S.J. *et al.* (2018) ‘Quantifying Two Dimensional (2D) and Three Dimensional (3D) Anatomical Learning Using a Neuroeducational Approach’, *The FASEB Journal*, 32(S1), p. 25.1-25.1. Available at: https://doi.org/10.1096/fasebj.2018.32.1_supplement.25.1.

- *Arduino Software (IDE) | Arduino* (no date). Available at: https://wiki-content.arduino.cc/en/Guide/Environment (Accessed: 20 May 2024).

- *Arduino vs ESP8266 vs ESP32 Microcontroller Comparison* (2019). Available at: https://diyi0t.com/technical-datasheet-microcontroller-comparison/ (Accessed: 20 May 2024).

- Badea, A., Kostopoulos, G.K. and Ioannides, A.A. (2003) ‘Surface visualization of electromagnetic brain activity’, *Journal of Neuroscience Methods*, 127(2), pp. 137–147. Available at: https://doi.org/10.1016/S0165-0270(03)00100-6.

- Birgani, P.M. and Ashtiyani, M. (2007) ‘Wireless Real-time Brain Mapping’, in F. Ibrahim et al. (eds) *3rd Kuala Lumpur International Conference on Biomedical Engineering 2006*. Berlin, Heidelberg: Springer, pp. 444–446. Available at: https://doi.org/10.1007/978-3-540-68017-8_112.

- Blinowski, G., Kamiński, M. and Wawer, D. (2014) ‘Trans3D: a free tool for dynamical visualization of EEG activity transmission in the brain’, *Computers in Biology and Medicine*, 51, pp. 214–222. Available at: https://doi.org/10.1016/j.compbiomed.2014.05.006.

- Brookes, M.J., Woolrich, M.W. and Barnes, G.R. (2012) ‘Measuring functional connectivity in MEG: A multivariate approach insensitive to linear source leakage’, *NeuroImage*, 63(2), pp. 910–920. Available at: https://doi.org/10.1016/j.neuroimage.2012.03.048.

- *Curry 8 | NEUROSPEC AG Research Neurosciences* (no date). Available at: https://www.neurospec.com/Products/Details/1031/curry-8 (Accessed: 20 May 2024).

- Eck, J. and Goebel, R., Esposito, F. Formisano, E. (2006) ‘Getting Started Guide Version 4.1 for BV 22.0’.

- ‘High-density EEG electrode placement | Robert Oostenveld’s blog’ (2006), 25 January. Available at: https://robertoostenveld.nl/electrode/ (Accessed: 2 August 2024).

- Jurcak, V., Tsuzuki, D. and Dan, I. (2007) ‘10/20, 10/10, and 10/5 systems revisited: Their validity as relative head-surface-based positioning systems’, *NeuroImage*, 34(4), pp. 1600–1611. Available at: https://doi.org/10.1016/j.neuroimage.2006.09.024.

- linda grefen (no date) *LED Lit Brain*. Available at: https://www.treatstock.com/3d-printable-models/3355223-led-lit-brain (Accessed: 5 August 2024).

- Liu, F., Dabbish, L. and Kaufman, G. (2017) ‘Can Biosignals be Expressive?: How Visualizations Affect Impression Formation from Shared Brain Activity’, *Proceedings of the ACM on Human-Computer Interaction*, 1(CSCW), pp. 1–21. Available at: https://doi.org/10.1145/3134706.

- Malik, A.S. *et al.* (2015) ‘EEG based evaluation of stereoscopic 3D displays for viewer discomfort’, *BioMedical Engineering OnLine*, 14(1), p. 21. Available at: https://doi.org/10.1186/s12938-015-0006-8.

- Mammone, N. *et al.* (2010) ‘Visualization and modelling of STLmax topographic brain activity maps’, *Journal of Neuroscience Methods*, 189(2), pp. 281–294. Available at: https://doi.org/10.1016/j.jneumeth.2010.03.027.

- Michel, C.M. and Murray, M.M. (2012) ‘Towards the utilization of EEG as a brain imaging tool’, *NeuroImage*, 61(2), pp. 371–385. Available at: https://doi.org/10.1016/j.neuroimage.2011.12.039.

- Pester, B. *et al.* (2022) ‘Understanding multi-modal brain network data: An immersive 3D visualization approach’, *Computers & Graphics*, 106, pp. 88–97. Available at: https://doi.org/10.1016/j.cag.2022.05.024.

- Sadiq, M.T. *et al.* (2022) ‘Exploiting pretrained CNN models for the development of an EEG-based robust BCI framework’, *Computers in Biology and Medicine*, 143, p. 105242. Available at: https://doi.org/10.1016/j.compbiomed.2022.105242.

- Sanmati Chaudhary (2018) *3D Light Up Brain Model*, *Hackster.io*. Available at: https://www.hackster.io/sanmati02/3d-light-up-brain-model-70d5df (Accessed: 5 August 2024).

- Shine, J.M. *et al.* (2019) ‘Human cognition involves the dynamic integration of neural activity and neuromodulatory systems’, *Nature Neuroscience*, 22(2), pp. 289–296. Available at: https://doi.org/10.1038/s41593-018-0312-0.

- TMSI (2022) *The 10-20 System for EEG*. Available at: https://info.tmsi.com/blog/the-10-20-system-for-eeg (Accessed: 2 August 2024).

- Vijayalakshmi, R. *et al.* (2014) ‘Change Detection and Visualization of Functional Brain Networks using EEG Data’, *Procedia Computer Science*, 29, pp. 672–682. Available at: https://doi.org/10.1016/j.procs.2014.05.060.

- Xiaoyan, W. *et al.* (2018) ‘Automatic seizure detection using three-dimensional CNN based on multi-channel EEG’, *BMC Medical Informatics and Decision Making*, 18. Available at: https://doi.org/10.1186/s12911-018-0693-8.

APPENDICES

Below is the code used for the testing of the LED strip to test its functionality.

Arduino IDE:

```
#include <Adafruit_DotStar.h>
#include <SPI.h>
#define NUMPIXELS 144
#define data_pin 9
#define clock_pin 10
Adafruit_DotStar strip(NUMPIXELS, data_pin, clock_pin, DOTSTAR_BRG);
uint32_t testcolor1=strip.Color(64, 0, 0);
  uint32_t testcolor2=strip.Color(0, 64, 0);
  uint32_t testcolor3=strip.Color(0, 0, 64);
//  uint16_t  = strip.numPixels();
byte m[5];

void setup() {
  // put your setup code here, to run once:
  Serial.begin(115200);
  strip.begin(); // Initialize pins for output
  strip.setBrightness(64);
  strip.show();

}

void loop() {
  // put your main code here, to run repeatedly:
  // if(Serial.available()>0){
  //  for (int i=0; i<3; i++){
  //   m[i] = Serial.read();


  //   if (m[i] == 1) {         //switch on the led
  //     strip.fill(testcolor1, 0,NUMPIXELS);
  //     strip.show();
  //     strip.clear();
  //   }
  //   if (m[i] == 0) {        //switch off the led
  //     strip.fill(testcolor2, 50,NUMPIXELS);
  //     strip.show();
  //     strip.clear();
  //   }
  //  }
  // }
//  for(int i=0;i<NUMPIXELS;i++){
//   if(i>=0 && i<=40){
//     int r=64;
//      int g=0;
//      int b=0;
//      strip.clear();
//      strip.setPixelColor(i, g, r, b);
//      strip.show();

//    }
//     if(i>40 && i<=NUMPIXELS){
//      int r=0;
//      int g=0;
//      int b=64;
//      strip.clear();
//      strip.setPixelColor(i, g, r, b);
//      strip.show();

//    }}
//   strip.clear();

 strip.fill(testcolor1, 0,NUMPIXELS); //fills the whole strip starting from 0th LED
with testcolor
  strip.show();
  delay(1000);
  strip.clear();
  strip.fill(testcolor2, 10,NUMPIXELS);
  strip.show();
  delay(1000);
  strip.clear();
  strip.fill(testcolor3,20,NUMPIXELS);
  strip.show();
  delay(1000);
  strip.clear();
//    // uint16_t n = strip.numPixels();
//    Serial.print(n);


//   strip.setPixelColor(5, 0, 255, 0);

// strip.setPixelColor(25, 255, 0, 0);
// strip.show();
// delay(2000);
// int t=millis();
// Serial.print(t);
// Serial.print("\n");
}
```

Below is the code for testing the serial communication between MATLAB and Arduino:

Arduino IDE:

```
String m;
void setup() {
  // put your setup code here, to run once:
Serial.begin(115200);
delay(2500);
}

void loop() {
  // put your main code here, to run repeatedly:
if(Serial.available()>0){
 m=Serial.readStringUntil("\n");
 Serial.print("Received: ");
 Serial.print(m);
}
}
```

MATLAB:

```
%fopen(instrfind);
%clc
%clear all
s=serialport('COM4',115200);    %to create the serial port in MATLAB
%fopen(s);                            %open the serial port
A= "Hi\n";             % A is a character
write(s,A,'char');
read(s, 2, 'char')
%fclose(instrfind);
```

Below is the code used for testing different baud rates to optimize speed of transmission between MATLAB and Arduino:

Arduino IDE:

```
#include <Adafruit_DotStar.h>
#include<SPI.h>
#define BAUDRATE 19200
#define NUMPIXELS 127
#define data_pin 9
#define clock_pin 10
Adafruit_DotStar strip(NUMPIXELS, data_pin, clock_pin, DOTSTAR_BRG);
#define NUMUSEFULPIXELS 50

int b;
int l;
uint8_t buff[NUMUSEFULPIXELS*3];
```

```
void setup() {
  // put your setup code here, to run once:
Serial.begin(BAUDRATE);
delay(1000);
strip.begin();
strip.show();
}

void loop() {
  // put your main code here, to run repeatedly:
if(Serial.available()>0){
    delay(NUMUSEFULPIXELS*3/BAUDRATE*1000);
    b = Serial.readBytes(buff,NUMUSEFULPIXELS*3);
    delay(NUMUSEFULPIXELS*3/BAUDRATE*1000);
 for(int i=0;i<b;i=i+3){
   strip.setPixelColor(i/3, buff[i],buff[i+1],buff[i+3]);
  }
  strip.show();
}}
```

MATLAB:

```
% test_leds_from_matlab

% definitions
serial_port_args = { "COM4", 19200};
 %serial_port_args = { "COM4", 115200};
Ncolours = 5;
Nleds = 50;

% derived
step = uint8( round( 255 / Ncolours));
% delay = 1 / Ncolours;
delay = 2;


%% tidy up any mess from before
if ~isempty( instrfind)
    fclose( instrfind);
    delete( instrfind);
end
if exist( 'sp', 'var')
    clear sp
end


%% set up the serial communications

% create the serial port object
sp = serialport( serial_port_args{ :});


%% run through a test procedure

% start at black, add a pixel at a time
```

```
fprintf( 'Black to blue\n');
colours = uint8( zeros( 3, Nleds));
pause( delay)
for i = 1:Ncolours
    colours( 2, :) = colours( 2, :) + step;
    Nbits = ledwrite( sp, colours( :));
    fprintf( ' %d %d\n', i, Nbits);
    %pause( delay);
end
pause( delay);

% turn off the leds
fprintf( '\n\nDone\n');
clear sp

return

% start at black, ramp to bright blue
fprintf( 'Black to blue\n');
colours = uint8( zeros( 3, Nleds));
write( sp, colours( :), 'uint8');
pause( 1)
for i = 1:Ncolours
    colours( 3, :) = colours( 3, :) + step;
    write( sp, colours( :), 'uint8');
    %fprintf( '%d ', i);
    pause( delay);
end
pause( 1);

% start at black, ramp to bright green
fprintf( '\n\nBlack to red\n');
colours = uint8( zeros( 3, Nleds));
write( sp, colours( :), 'uint8');
for i = 1:Ncolours
    colours( 2, :) = colours( 2, :) + step;
    write( sp, colours( :), 'uint8');
   % fprintf( '%d ', i);
    pause( delay);
end
pause( 1);

% start at black, ramp to bright red
fprintf( '\n\nBlack to green\n');
colours = uint8( zeros( 3, Nleds));
write( sp, colours( :), 'uint8');
for i = 1:Ncolours
    colours( 1, :) = colours( 1, :) + step;
    write( sp, colours( :), 'uint8');
  %  fprintf( '%d ', i);
    pause( delay);
end
pause( 1);

% turn off the leds
fprintf( '\n\nDone\n');
clear sp


function Nbits = ledwrite( sp, colours)

write( sp, colours( :), 'uint8');
```

```
Nbits = read( sp, 1, 'uint8');
```

Below is the code for converting EEG data to LED data for visualisation:

Arduino IDE:

```
#include <Adafruit_DotStar.h>
#include<SPI.h>
#define BAUDRATE 19200
#define NUMPIXELS 127
#define data_pin 9
#define clock_pin 10
Adafruit_DotStar strip(NUMPIXELS, data_pin, clock_pin, DOTSTAR_BRG);
#define NUMUSEFULPIXELS 50

int b;
int l;
uint8_t buff[NUMUSEFULPIXELS*3];
void setup() {
  // put your setup code here, to run once:
Serial.begin(BAUDRATE);
delay(1000);
strip.begin();
strip.show();
}

void loop() {
  // put your main code here, to run repeatedly:
if(Serial.available()>0){
    delay(NUMUSEFULPIXELS*3/BAUDRATE*1000);
    b = Serial.readBytes(buff,NUMUSEFULPIXELS*3);
    delay(NUMUSEFULPIXELS*3/BAUDRATE*1000);
 for(int i=0;i<b;i=i+3){
   strip.setPixelColor(i/3, buff[i],buff[i+1],buff[i+3]);
  }
  strip.show();
}}
```

MATLAB:

```
%% explore lighting up leds from matlab via an arduino

% definitions
participant = '0113';
task = 'study.fingertapping.all';
time_step = 0.1;
max_rms = 5e2;
serial_port_args = { "COM4", 19200};
count=0;
% serial_port_args = { "COM4", 115200, 'Tag', "Arduino"};
Nchan = 50;
```

```
delay = 2;


%% tidy up any mess from before
if ~isempty( instrfind)
    fclose( instrfind);
    delete( instrfind);
end


%% load the data

% load the data and get the start and stop times for the data
d = eeglocal.gamma.loadSubject( participant, task);
d = d.selectchan( 1:Nchan);
time_base = d.timebase;


%% set up the serial communications

% create the serial port object
sp = serialport( serial_port_args{ :});
pause(delay);

%% send data to the leds

% start at the beginning of the data
start_time = time_base( 1);

% loop sliding a block at a time until we go past the end of the data
while start_time + time_step <= time_base( 2)
    % select the data we want to process in this block
    r = d.selecttime( start_time, start_time + time_step).rms.data;
    % work out the colours for all the leds
    colours = uint8( r / max_rms * [ 255 255 255])';
    %changing the format of data to 1xn array
    %colours = reshape(colours, 1, []);
    % send the information to the arduino led driver
    write( sp, colours( :), 'uint8')
   % writeline(sp,"Do you see me?");
    %data = read(sp,1,'uint8')
   % Nbits = read( sp, 1, 'uint8');
   % fprintf( '%d\n', Nbits);
    % slide along for the next block
    start_time = start_time + time_step;
   %pause(0.5)
    % draw something on this screen to see if it is working
     image( colours);
    %count=count+1;
     drawnow;
end
pause(delay)
% and tidy up
%count
%data = read(sp,1,'uint8')
delete( sp);
```