Controlling a Robotic Hand Using a Real-Time Neurorobotic System

Emanuel Bjurhager*, Simon Friberg†, Joakim Nylander Nordström‡, Karl Söderlund§,
Inas Hamzic¶, Sofia Blaad‖, Jonathan Holm**, Pontus Gustavsson††, Hans Tschohl‡‡, Jonathan Fernkvist[x]
Fredrik Åbrink[xi] Ludwig Wässman[xii] Oliver Lagelius[xiii] Victor Aziz[xiv]
School of Innovation, Design and Engineering, M.Sc.Eng Robotics
Mälardalens University, Västerås, Sweden
Email: [*ebr17003, †sfg18002, ‡jnm18003, §ksd18001, ¶ihc18002, ‖sbd18004, **jhm18006, ††pgn18006,
‡‡htl21001, [x]jft18002, [xi]fak1800 [xii]lwn15002 [xiii]ols16001 [xiv]vaz17001

***Abstract*—**

A. Component 2A - Inas

This paper will cover the implementation of a real-time neurorobotic system to control, using the mind, the gripping of an object by a robotic hand. This was done by dividing the system into three components, where each component had different tasks. The first component handles the signal acquisition, which means collecting data and streaming it. The second component focus was signal processing, meaning processing the signal into a control command for the robotic hand. The last component tasks were controlling, as well as optimizing, the robotic hand. This paper thoroughly describes each component in detail, including; the software and hardware used, different processing methods, but also optimization techniques.

B. Component 2B - Jonathan F

A few different filters were tested to find the best one suited for the project. Common Spatial Patterns (CSP) was used to extract features from the data, which was then classified using Support Vector Machine (SVM) where the accuracy was around 60%. Improvements can be done by sampling more data with better quality, a better filter and optimizing the CSP and SVM.

***Index Terms*—BCI, CSP, EEG, LSL, Robot Hand, SVM**

I. Introduction

A. Component 2B - Sofia

Patients with disabilities in motor response movements of their limbs or neuromuscular disorders can be helped through a Brain Computer Interface (BCI). BCI is a communication system that interprets brain signals which later results in a command for an external device. This external device is not connected to the user through muscles or nerves. But instead, the commands are sent by brain activity. Hence no motor abilities are needed, it is controlled through thoughts [1][2][3].

Receiving brain activity can be done in invasive and non-invasive ways. In this report, the signal is obtained through the non-invasive Electroencephalogram (EEG). The amplitude of the electrical impulses is measured by electrodes that are placed on the scalp of the head. EEG captures a temporal resolution of less than a millisecond when examining the cortical activity. Different brain activity gives distinct traces. EEG signals are random, non-linear, non-gaussian and non-correlated [1].

The EEG signal has a low signal-to-noise ratio which lets background noise invoke the readings of the brain activity. These artifacts can be eye-blinking, muscle activities and breathing. The characteristic of the signal also depends on the individual, the mental state, and the age of the user [1].

Typically, the frequency ranges between 1-100 Hz and the amplitude is between 10μV and 100μV. The frequency can be divided up into different frequency bands which can be seen in Table I [1].

Table I
Brain rhythms

Rhythm	Frequency Range	Amplitude
Delta	<4 Hz	20-200 μV
Theta	4-8 Hz	>20 μV
Alpha	8-13 Hz	30-50 μV
Beta	13-30 Hz	5-30 μV
Gamma	>31 Hz	<5 μV

For the BCI created in this report Motor-Imagery (MI) EEG is used. MI is a cognitive process that lets the user imagine a movement without physically performing the task. The MI will induce a change in the EEG signal which will be used to control the BCI [2].

B. Component 2A - Victor

The source of these changes in neural activity for MI tasks is known to be located in the motor cortex. Different regions of the motor cortex are associated with different body part movements.

Even if the changes in the neural activity and their locations on the motor cortex is known, A non-invasive Brain Compute interface faces a lot of challenges on how to measure that neural activity and detect the changes. Apart from the low signal-to-noise ratio stated above the spatial characteristic of the brain signal leads to a volume conduction problem. The neural activity in the motor cortex is conducted through the brain volume to the scalp and into the sensors. This means that the data collected by a sensor at a specific location does not necessarily contain data from the source corresponding to that location. Each electrode measures a weighted sum of each neuron activity and a combination of noise and artifacts.

A BCI design consists of different temporal and spatial filters techniques used to extract those changes. The extracted feature is applied to a classification model that classifies those features into different movements.

C. Component 3 - Joakim

After receiving and interpreting Motor-Imagery data, it can then be used to control some mechanical object. In this project, that is a robotic hand. It mainly consists of 3D print, servos, RoboRIO board, and LabVIEW software. The cooperation between these gives a potential interface that involves human interaction and learning for medical treatment. However, this project does not include that specific type of feedback training, but rather an unbiased thought process. That is because subjects in this study have no hand motor impairment. Brain activity data collected from this study should rather be seen as a basis that allows for robotic hand feedback training with disabled patients.

D. Hypothesis

E. Component 1A - Oliver

The aim of the project for component 1A will be to collect and to stream EEG data in real-time as structured packages to component 2A, as well as creating a task protocol that are synchronized with the streamed data. The streamed EEG data also needs to be saved, such that it can be analyzed offline.

F. Component 1B - Pontus G

For component 1B the project will have it's focus on sampling EEG data and creating a protocol. The system needs to be able to store offline data into text file, being able to represent and stream the data online in real-time as well as having an easy to follow protocol.

G. Component 2A -

H. Component 2B - Sofia

The project aims to create a real-time neurorobotic system by building a BCI that controls a robotic hand by imagining relaxing or closing your hand. EEG collects brain signals from the patient who is following a protocol. Then the signal will be pre-processed, the features extracted and then classified into a command in real-time. The command (closed or relaxed) will be executed by the robotic hand.

Following the introduction, the background will be presented and the methods for each component will be introduced and then the results, respectively. Lastly comes the discussion and conclusion that the components have drawn.

I. Component 3 - Joakim

Regarding the mechanical steering of the robotic hand, the first and main goal is to use all of its fingers to represent "Open"/"Relaxed" and "Closed" states. The later state with hope of gripping objects. This is the main goal because it is the request from component 1 and 2. The second goal is to make it possible to not only open or close it, but also to move individual fingers. Additionally, a supplementary state to the two states "Open"/"Relaxed" and "Closed" is desirable. These extra states have the purpose of assisting in further research of potentially more than two hand states.

II. Background

A. User Datagram Protocol - Simon

For transferring data between components a protocol has to be used. The protocols that were looked into in this project were the User Datagram Protocol (UDP) and the Transmission Control Protocol (TCP). TCP was looked at due to it's guarantee on packets arriving in order, while UDP was looked at due to it's low latency. According to tests ran by Madhuri and Chenna [4], the End-to-End delay of UDP was quite a bit better than that of TCP, while TCP had a much higher packet delivery ratio. These tests show however that the packet delivery ratio of UDP is still quite high at the lower data rates that are used in this project.

B. Data interpolation (Inas)

Interpolation is the process of estimating unknown values, by using known values. There are multiples different techniques of interpolating, the most common being linear interpolation. This technique uses knowledge of two data points, and calculates the rate of change between them, which is assumed to be constant. To calculate this, a slope formula is used. Using the rate of change between two points, one can interpolate any values between them. [5]

C. Classification algorithms (Inas)

Classification is the process of recognizing objects into preset categories. Machine learning algorithms for classification use training data as input to predict the categories for future data. There are different type of algorithms for classification, specifically:

1) Linear-Discriminant Analysis: This type of algorithm is fast and straight-forward, it is mostly used to take a high dimension of features, and reduce them into a lower dimension [6]. This is done by finding a hyperplane which minimizes the variance between two classes, while maximizing the mean distance between them [7].

2) Naive Bayesian: This algorithm is supervised, and is based on Bayes' Theorem, meaning it is a probabilistic classifier which assigns features to the class which has the highest probability of belonging [8]. As the algorithm is based on Bayes' Theorem, it is assumed that that all features in every class is independent and normally distributed [6].

3) Support Vector Machines: Support vector machines (SVMs), is a powerful supervised machine learning algorithm, which can be used in regression and classification; although it is mostly used in classification [9]. The algorithm is mostly used when classifying two classes, this is done by finding the best hyperplane that separates data between the two classes. This means, the model which has the largest margin between the two classes is the best [10].

D. Validation algorithms (Inas)

There are different algorithms related to validating a machine learning model, which will be explained briefly:

1) Train/test split: This method for testing and validating data is the most basic, the data is split randomly into 70% for training and 30% for testing the model. This is good, as it shows how well the model reacts to previously unseen data. Due to this type of split being a non-random sample of a population, it is referred to a model with a sampling bias. This is due to some members having a higher chance to be selected than others. [11]

2) k-Fold Cross-Validation: This type of technique minimizes sampling bias by making multiple splits, then validating the data on all the different splits. This is done by, first, splitting the data into k-folds, see fig.**??**. Further, the data is trained on k-1 folds and then tested on the fold that is over. This is done for all the different possible combinations, then takes the average of the result. By using this technique, all observations are used for validation and training, which minimizes sampling bias. [12]

3) Nested Cross-Validation: By using this method, the hyperparameter tuning steps are separated from the error estimation step. This is done by nesting two k-fold cross-validation loops, where the outer loop is used for estimating accuracy, and inner loop for hyperparameter tuning. By splitting up the model estimating accuracy, and the model for the best hyperparameter, the risk of overfitting is decreased. [13]

E. Lab streaming layer(LSL) - Jonathan H

Lab Streaming Layer (LSL) is a protocol for streaming of data. The protocol have advantages over other protocols when it comes to sending of EEG data. LSL gives researchers great precision when it both comes to timing and synchronize data recorded in different sampling rates. LSL also gives the possibility to receive data from multiple streams at the same time, this is through the use of ping. One of the large selling point is the possibility of real time processing of the data. Interfaces for many language such as MATLAB[14] can easily be setup. LSL can also be configured to suit the need for the application area. where options as send samples one at the time or by sending chunks can be configured. Built in the protocol is also a intuitive sending protocol provided; where the receiver of the data can simply see each of the channels (if multiple was used during the recording)[15].

F. Component 2 - Sofia - Victor

1) Data analysis: In the EEG data analysis, the observed features in the signal are the changes in the oscillatory processes like the alpha rhythm (8-13 Hz). While the Fourier transform is the main converting method to the frequency domain, the results cannot be easily interpretable for non-stationary signals like the EEG signals. The Fourier transform of a data window will show the frequency content of that window despite how much each frequency is appearing in that window. Smother results may be observed by estimating the spectral density of the EEG data in certain frequency intervals. There exist different approaches for that estimation like Welch's method. The method consists of dividing the signal into windows and computing the Fourier transform of each window and averaging the results. There are still two problems with applying this method on EEG data firstly the frequencies represented by the result are limited to the size of the window chosen. Secondly, the method does not represent the temporal dynamics of the signal[16].

The BCI in this paper is based on the changes of the amplitude in certain frequencies over time, this is best observed by using a frequency analysis technique that preserves the time information. The most used techniques are the Hilbert transform and the wavelet analysis. The results of both methods may be similar but the Hilbert transform works best on narrow frequency band data. It is recommended to filter the data to a narrow band before applying the Hilbert transform[17].

The wavelet transform uses a kernel called wavelet to extract the frequency and time information from the signal. The type of wavelet is data specific and each type of data has a commonly used type of wavelet. For the EEG data, a commonly used wavelet is the complex Morlet wavelet which is represented by a Gaussian window in the frequency domain. A commonly used algorithm in wavelet analysis is the multi-resolution wavelet analysis that computes the wavelet transform in multiple cycles while varying the wavelet frequencies, this is useful to solve the problem of the trade-off between time and frequency resolution.[16]

2) Data Processing: The goal of the processing is to separate the sources of the changes in EEG signal from the rest of the data. Multiple techniques can be used for this purpose, the methods mentioned in this paper are spatial filters that act as dimension reduction of the data. The filter is the coefficients of the weighted combination of each channel. The mixture of the weighted combination of each channel results in a component that corresponds to the desired source of the neural activity separated from the data. To apply the filter the data is multiplied by the filter matrix which projects the data in the desired direction and extracts the relevant source[18].

For oscillatory processes, common spatial source separation is the Independent component analysis (ICA). It is a blind separation technique that is based on the non-gausienity of the sources. Neurological oscillations tend to be Gaussian distributed which makes ICA limited to be used for separation of the neurological sources. On the other hand, other artifacts and noise in EEG data may not be Gaussian distributed which makes ICA useful to separate those types of artifacts from the data, for example, eye blinks and eyes movements.

Lastly one of the most popular spatial filters used in EEG data is the Common Spatial Pattern (CSP). The CSP method can be described as a generalized PCA, as it will find directions in the multivariate data which maximize the difference between two classes. It does that by finding directions that maximize the variance of one class while minimizing the variance of the other class.

The problem with CSP is that it can easily overfit the data

as the difference between the two classes may be noise that exists only in the training data. A solution to that is to train the filter on independent data that is not related to how the filter is used in the system. There exist a lot of different CSP algorithms and extensions that are useful in different types of applications, some of them offer adaptive frequency bands other have multiple time and frequency windows.

The computation of the CSP can be done in different ways. The first one is that it can be computed as a Generalized Eigenvalue Decomposition (GED). It can be done by computing the eigenvectors of the ratio between the two classes. To avoid overfitting the computation of eigenvectors can be done between one class and the average between the two classes. The results are the directions in the data in which the used class differs from the average data. Another way to compute that approach is a geometric method that is done in three steps. The first step is to compute the whitening transform for the average data. Secondly, apply the transform calculated in the first step on one class. The third step is to compute the PCA for that class. The CSP filter is then applied by applying the whitening transform and projecting the data in the PCA component's directions[17].

3) Related Work: When studying rehabilitation of motor ability there may be a challenge to separate EEG patterns of a MI in the same limb. One article [19] conducts a study involving gripping and extension of fingers. They show that these two classes cannot be distinguished when based on EEG spectral power. A reason being that the cases involve comparable muscle groups resulting in activating the same area of the brain which makes it difficult to separate them linearly.

In [2], MI was used to classify between left-hand and right-hand imaginary motions. A decrease was observed in the signal when MI was performed, this being Event-Related Desynchronization (ERD). When the process of imagining was finished the signal would retain, then Event-Related Synchronization (ERS) occurred. Both ERD and ERS present themselves in the frequency bands of alpha and beta. A change over the sensorimotor area can be found when performing MI by moving the right and left hand. During both imagining and preparing for a hand movement, a change in ERD can be detected [20]. Imagining a movement shows a decrease in alpha and beta band [19].

In component two the signal needs to be pre-processed. The artifacts will be removed by the use of a filter. Then the features will be extracted. To do this one can use both linear and non-linear techniques. Moreover, the signal can be analyzed in different domains. Brain states can be extracted using Common Spatial Pattern (CSP) [1].

Constructing a hybrid BCI has been shown to provide better results than single-model BCI [21]. This article [21] implements both MI and P300 in their study about cursor control. Their pre-processing consists of a band-pass filter with cut off frequencies of 0.5 and 100Hz when sampling at 250Hz. For their P300 they use Support Vector Machine (SVM) to classify the feature vector from concatenating the channels. The score output represents the confidence level and is used along with a threshold to decide if the P300 signal is detected or not. For the mental activity, they first filtered using Common Average Reference (CAR). After that, a band-pass filter for 8-14Hz is applied. To extract the features CSP was used. Those features extracted were the input to the SVM. The CSP filter is trained using 60 trials of right- and left-hand MI tasks. Using only MI it gave an average accuracy on the classification using 10-fold cross-validation of nearly 80%. Standard deviation was also calculated to be around $\pm$ 5 percentage points.

Classification is the last step of real-time data interpretation. One review [3] compares different classification algorithms in different situations. The situations varied in terms of feature extraction method, pre-processing and EEG source. They compare the average performance and accuracy on Linear Discriminant Analysis (LDA) and SVM among other classifiers. LDA is a popular algorithm in BCI since it is straight forward and is not affected by small variations leading to a stable classifier. SVM, on the other hand, can be both linear and non-linear depending on the kernel function chosen. The SVM presented small errors in different BCI systems. Gaussian SVM is mostly used and compared to LDA, it is hard to overtrain. Also, when comparing the two classifiers SVM shows better results. This is since it is insensitive and tolerant.

A third article [22] estimates the strength of MI when closing and opening a hand by extending the fingers. Then an FIR band-pass filter was used between 1-100Hz while sampling at a rate of 1kHz. They then extracted the features using Power Spectral Density (PSD) in a window of 500ms for each trial. Later Hierarchical Genetic Algorithm (HGA) was used to select some features. A linear kernel SVM was used to discriminate the MI. Using only the 5% of the highest ranked in the HGA, the SVM accuracy showed the highest result of $77\% \pm 1.7\%$.

G. Component 1A - Ludwig W

The quality of all machine learning and data classification relies on the foundation of the data quality. As mentioned previously in the background section, machine learning is a technique to train computers how to make accurate and precise predictions based on previous experiences with a data set. However in order for the computer to actually make correct predictions it has to be exposed to data that contains the desired information.

For this project, and especially for this project component, the acquisition of useful data in an applicable data size is one of the major challenges. If component one feeds insubstantial data forward to component two the machine learning algorithms will not be able to classify user intent properly, thus the system will fail.

There are two major risk factors that component one has identified in before hand. The first is that the technique used to acquire the data have to be correct, resulting in a signal with a high SNR. Second the transmission of data is of risk to lose packages resulting in information loss.

H. Component 1B - Hans

The first step to creating this BCI was to acquire data from subjects in real time. This can be done using either invasive or non-invasive methods.

Invasive methods such as Electrocorticography (ECoG) require implants beneath the skull directly on the surface of the brain to receive information and record the electrical activity from the brain. [23]

Non-invasive methods include Electroencephalogram (EEG) or MEG. The EEG method makes use of a headpiece which contains several electrodes that will be placed directly on your sculp to measure the potentials between each electrode placed on the head and one reference electrode which is ideally placed on the earlobe of the subject. [24]

The method of streaming the data in real time is important and can vary depending on the function of the BCI. The biggest two Internet Protocols (IP) to choose from are Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). UDP is known for being faster and would be a great choice for BCIs that require instantaneous reaction time such as prosthetic arms. TCP is connection orientated, which means that once it has been connected to a BCI, it will be able to receive feedback from the computer to the microcontroller on the headpiece. The main disadvantage to TCP is that it is slower than UDP. [25]

I. Component 3 - Emanuel

A similar hand was built by S. Park Et Al[26]. It describes a similar hand, but with 4 fingers and 16 degrees of freedom, compared to our component that has 5 fingers and 5 degrees of freedom. Their approach yielded much better control of the hand, and they were able to move the fingers in ways that are physically impossible with our hand. Another study by R. Fourie Et Al[27] shows a hand that more closely resembles our hand since they also only use 5 degrees of freedom, but their thumb was in such a way that holding an object should be easier. There is also a study by B. Gámez Et Al [28] where only three degrees of freedom were used, they similarly to us, also use a very simple control protocol. In their protocol, only open hand, closed hand, pointing with index finger and precision grasp using index finger and thumb are possible. This more closely resembles our hand, but they have the advantage of a better thumb placement while we have the advantage of individual finger control. All these studies placed the motors or servos inside the hand while we place them in the forearm. An advantage of our approach is that we are not as limited by space.

J. Component 3 - Joakim

From the previous year in the *Neurotechnology* course that this project is a part of, a 3D-printed hand was used. This consisted of palm and forearm as a stable basis, and three-joint fingers simultaneously moved by Parallax servos [29] placed inside the forearm, connected with fishing lines. These servos were connected to RoboRIO [30], a device capable of sending Pulse Width Modulation (PWM) signals to control the speed of each servo. The Parallax servo however misses the ability to accurately align itself to keep a certain position at an angle, and thus has limited possibility of representing hand states. In addition, it is insufficiently strong to grip objects when still. An alternative to the Parallax servo was Hitec servo [31] that is also controlled by PWM, but in this case its characteristics (duty cycle) is used to hold a specific state, stronger than a Parallax servo.

LabVIEW is one program available for RoboRIO. There is a LabVIEW software specifically made for RoboRIO [32], the version of this that was available for this project is the 2016 version. The software enables the user to program with a visual, block diagram-like, interface. There are specific blocks with commands that can be used to send out a PWM signal with the RoboRIO board. That is what students of the previous year used during their project, with the same goal of moving a robotic hand.

III. Method Component 1A

A. Hardware 1A Fredrik

The Ultracortex mark IV designed by OpenBCI headset was used to measure the EEG signal. The complete kit from OpenBCI contains a 3D printed headset, a cyton board together with the daisy module, an USB dongle and TDE-200 dry EEG electrodes [33][34][35].

The headset allows placement of up to 35 electrodes different node positions that follow standardized 10-20 configuration system (see fig 1. Initially in the project, 16 electrodes were used to measure brain activity, but due to broken wires 3 electrodes rendered no usable data. The OpenBCI GUI did not label the electrodes according to std 10-20, table II shows how This projects electrode placement seen in fig 2 correlates to std 10-20 placement.

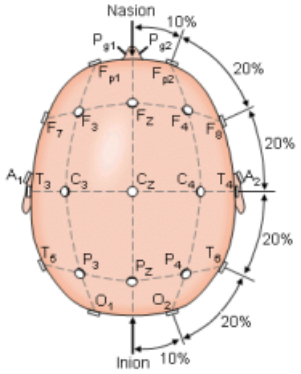


Figure 1. The image shows the electrode placement according to the standardized 10-20 configuration system. [36]

Project electrode labels and corresponding 10-20 labeling					
Proj.	10-20	Proj.	10-20	Proj.	10-20
1	F_{p1}	7	O_1	13	P_4
2	F_{p2}	8	C_Z	14	P_Z
3	C_3	9	F_3	15	T_5
4	C_4	10	F_4	16	T_6
5	T_3	11	O_2	Left ear	A_1
6	T_4	12	P_3	Right ear	A_2

Table II
THIS PROJECT VS STANDARDIZED 10-20 ELECTRODE PLACEMENTS

The measured data was sent via the Bluetooth 4.0 wireless communication protocol to the OpenBCI USB dongle receiver [37].

The TDE-200 is a dry comb electrode that is made of Silver-silver chloride (Ag/AgCl) with a diameter of 10 mm. They are designed 12 5mm long pins to improve contact through the hair and provide transmission of surface biopotentials. The electrodes are used dry and do not require the addition of a gel to improve conductivity. Flat dry electrodes are used for the forehead. i.e 1 (F_p1) and 2 (F_p2). [38][35][39].

The OpenBCI Cyton evaluation board is an EEG amplifier built around the ADS1299 biosignal amplifier. The ADS1299 records up to 8 DC-coupled (which is extended to 16 with the OpenBCI daisy module) differential biopotentials with 24-bit digitization. Digitized samples from the ADS1299 are received by a PIC32MX250F128B microcontroller [40] with a chipKIT UDB32-MX2-DIP bootloader. The microcontroller runs at 8 MHz with 32 kb of memory and uses a serial communication bus to transmit data to a microcontroller called the RFDuino (BLE radio). The cyton board is also equiped with a LIS3DH 3 axis accelerometer to provide data on the x- y- and z-axis. [39][41].

B. Data Acquisition 1A - Oliver

There are two different states that the subject may consciously switch between. These states are (i) relaxed hand and (ii) closing the hand. The theory is that relaxing is an idle task while closing is the working task. The test conditions where that the subject was positioned in a shielded, dark, noiseless room, visually looking on a white dot on a black wall. The signal was an audio signal for timing.

Protocol [One session]	
State	Time [sec]
Relaxed	10
Closed	4
Relaxed	4
Closed	4
Relaxed	4
Closed	4
Relaxed	4

The software used to obtain the data was OpenBCI GUI. The data is logged with a frequency of 125Hz to a txt file. Each sample consists of 16-channels of neural data, accelerometer data in the x-, y- & z-plane and timestamps in format [hh:mm:ss:ms].

The project used two subjects with a total of 15 sessions each. This is equivalent to ten minutes of recording per subject i.e twenty minutes in total. The electrodes measure the electrical potential on the surface of the subjects scalp which correlates with the electrical potential over the neurons.

C. Physical setup - Oliver

To Measure the neural activity a non abrasive method was used. Sixteen electrodes was strategically placed on the subjects head. The impedance threshold over the electrodes was 250kΩ or lower.

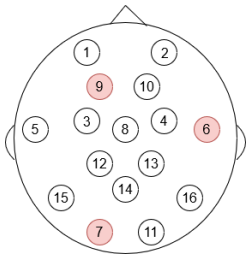


Figure 2. Placement of all electrodes on the subjects scalp. Electrode 6, 7 & 9 where disregarded due to malfunction.

D. Real Time configuration - component 1A - Ludwig W

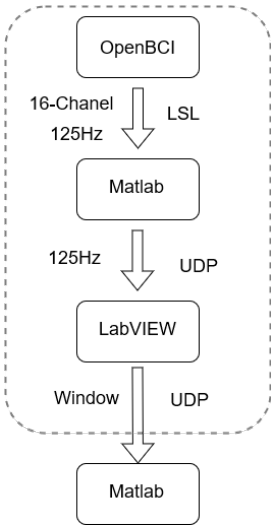


Figure 3. Flowchart of the system configuration for component 1A. The data is obtained from the OpenBCI Graphical User Interface and then it is transferred at a rate of 125Hz via lab streaming layer (LSL) to Matlab in the format of float variables. The Matlab program transforms the LSL stream in to a string UDP stream (125Hz) and transfers the data to Labview. Labview is responsible for saving the data in windows which are then sent with an interval via UDP to matlab, accessible for component 2A. The window size is set to 100 samples.

1) Data type: each sample of the data is represented by one string containing 16-channels of data and x- y- and z-accelerometer data.

2) Lab Streaming Layer: is a data communication protocol that enables synchronized and streamlined acquisition of time series measurements across multiple devices. This project is

using the inbuilt LSL protocol in OpenBCI to stream the data, However since receiving data from LSL in Labview is difficult this project chose to translate the LSL stream in to a UDP stream through Matlab.

3) User Datagram Protocol: is a simple connectionless data communication model. It is a protocol that sends one message at a time without expecting acknowledgments, it allows the system to work fast with the risk that packages may be lost. This communication protocol was chosen based on the supposition that speed would be more important than reliability for this system.

IV. Method Component 2A

A. Component 2A - Offline

1) Data analysis - Inas: Two different approaches of data analysis was done, to see if the different approaches yields the same result. The first step of data analysis in offline was to analyze all recordings to make sure they are compatible for processing, this was done by manually analyzing the signal and the frequency spectra of the signal. If too much noise was active, it was noted in a spreadsheet. At first, the entire recording was examined at once. It was noticed that plenty of recordings had an unnatural frequency spectra, see fig. 7. By further inspection, it was noted that the pulse causing this type of frequency spectra is at the start of the recording. Hence, a decision was made to remove the first ten seconds of all recordings, and split the rest of the recording in the following windows:

- ***Grip 1***: 10 - 14 seconds
- ***Relax 1***: 14 - 18 seconds
- ***Grip 2***: 18 - 22 seconds
- ***Relax 2***: 22 - 26 seconds
- ***Grip 3***: 26 - 30 seconds
- ***Relax 3***: 30 - 34 seconds
- ***Grip 4***: 34 - 38 seconds
- ***Relax 4***: 38 - 42 seconds

By doing this, analyzing the different intervals will give a clearer view as noise present in other signals will not affect the rest of the intervals. After splitting the signal, all 16 channels of the signal were thoroughly analyzed in all the intervals. As mentioned above, the signal in time domain and frequency spectra was analyzed; if there exists too much noise, see fig.5 and fig.6, the interval and channel which the disturbance occurs at was noted on a spreadsheet.

The next part of data analysis is filtering. The frequencies of which contains relevant information are spread around 10 Hertz, which means a filter which can remove irrelevant noise was designed. To achieve this, different filtering techniques were applied and tested. While testing different filters and parameters, one of the key factors which was important was to note how much of the relevant information was lost while filtering. This is important, as losing too much of the wanted signal could affect the processing and classification part. As mentioned above, the frequency from brain activity is around 10 Hertz, thus different bandpass filters around that range was tested. The filter which gave the best results was a FIR bandpass filter between the ranges 0.5 and 30 Hertz, see fig.4.

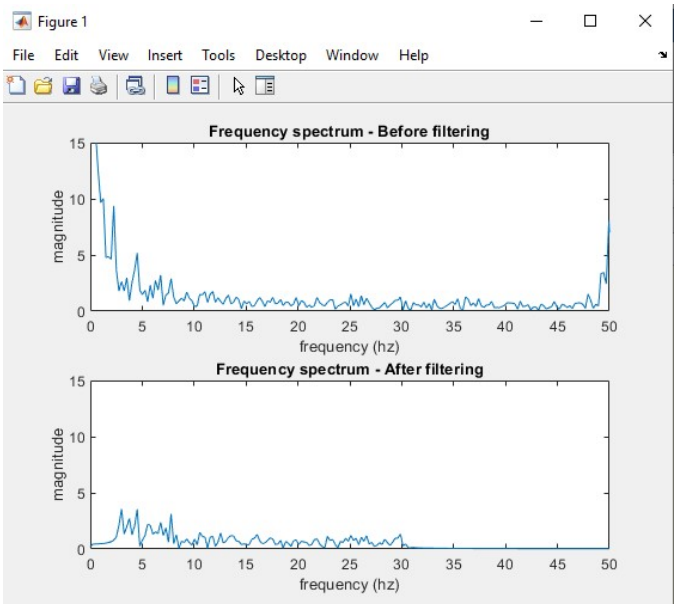


Figure 4. Frequency spectra of one signal, before and after filtering with a bandpass between ranges 0.5-30 Hertz.

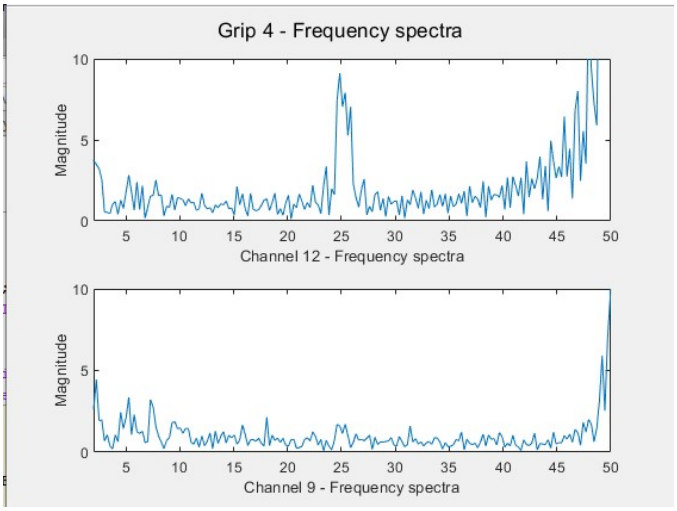


Figure 5. Two different plots of different signals in frequency spectra; the signal above showing how a noisy channel looks, while the bottom shows a clear signal.

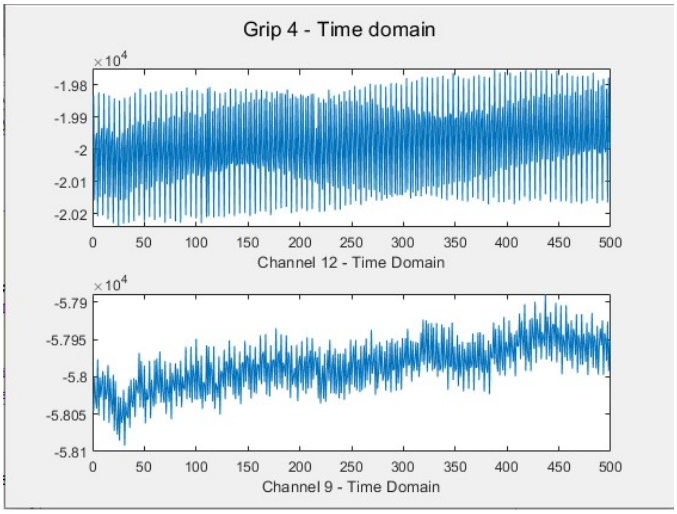


Figure 6. Two different plots of different signals in time domain; the signal above showing how a noisy channel looks, while the bottom shows a clear signal.

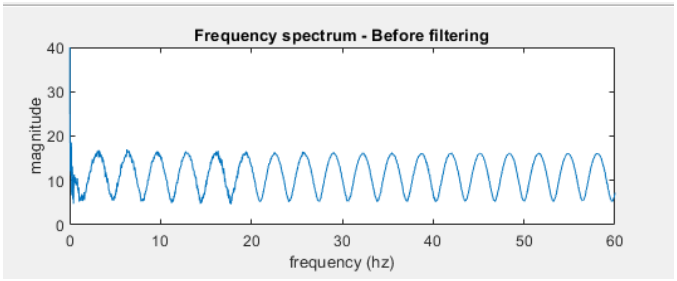


Figure 7. Frequency spectra of one of the recording, showing the unnatural frequencies.

2) Data analysis - Victor: The data acquired by component 1A is divided into trials where each trial represent an IM relaxed hand or IM grip hand. The data from each trial is plotted in the time domain and compared to a filtered version of the data. The filter used is a FIR band-pass filter between 0.5-30Hz designed using the least-squared linear-phase method. The goal of the comparison is to detect bad channels and bad trials. Based on the result of this comparison the component acquiring the data was notified about the bad channels and bad trials found. For further analysis of the data the component acquiring the data needed to acquire new data, which was not accomplished because of hardware problems and time limitations.

While waiting for component 1 to acquire some data the algorithm implemented needed to be verified on other data. It was not possible to find EEG labeled data with the same classes Grip and Relax. The data used in the rest of this method is the calibration data of the first dataset of the Berlin BCI competition 4 from one person. The data consist of 2 classes 100 epochs of IM left-hand movement and 100 epochs of IM right-hand movement. The data is acquired by 59 channels with a 100Hz sampling frequency. The recording is done by [42].

The data is analyzed per channel using a complex morlet wavelet with the method mentioned in [17] the baseline is taken from -400 ms to -100ms from the cue at time 0. The goal of the analysis is to observe a suppression of the frequency around 8 Hz to 13 Hz while comparing the time-frequency domain of channel C3 to channel C4. The time in which those suppression happen is also observed for extracting the relevant data windows.

3) Data Windows - Victor: The data is divided into 60 % training set and 40% test set. The relevant data in the window 500 ms to 2500ms is extracted from the epochs. The extracted data is divided into windows based on the lecture [43]. The extracted data from the epochs is divided into windows of 1000ms while overlapping by 800ms.

4) Temporal filter - Victor: The windows in training data are processed to calculate a spatial filter following the CSP method from[44]. Each window is bandpass filtered between 8Hz and 15 Hz using a 20 order FIR filter with linear phase designed using the least-squared method. In matlab this is done using the function firls().

5) Spatial filter filter - Victor: The next step is to calculate the covariance matrix of each class. For that, the data need to be normalized with a zero-mean normalization. The average covariance matrix of each class is calculated by multiplying the window with its transpose and averaging over all the training set for each class. Each class has an average covariance matrix of the size of (59 x59) that corresponds to the number of channels. To observe the covariance matrix of the classes the function imagesv() is used.

To calculate the whitening matrix PCA is applied to the sum of the average matrices of both classes.

In MATLAB PCA is applied by using the function svd() which calculates the eigenvalues E and eigenvectors V of the data. The whitening matrix P is calculated by the matrix multiplication in equation 1.

$$P = V \times (E)^{-\frac{1}{2}} \tag{1}$$

S_1 is the result of applying the whitening matrix on the average covariance matrix of one class (Left-hand movement class). Another PCA decomposition is calculated using svd() for S_1 . It results in another set of eigenvectors R. The spatial filter W is to apply the whitening matrix and then project the data on R as shown in 2.

$$W = R \times P \tag{2}$$

The spatial filter used is the first and last component of W. Both the training data and test data are bandpass filtered and spatially filtered by the training spatial filter W which reduces the dimension of the data from 59 to 2. The feature extracted is the logarithm of the variance of the data which is classified in the next part.

6) Classification - Simon & Inas: For classification, the machine learning method chosen was Support Vector Machine (SVM), which is also one of the methods used in [6]. From the processing part, a training set with 720 trails and a testing set with 480 trials was received, containing two features. When

training the SVM, two parameters are needed; one parameter containing all the features, and the other containing the classes in order. The parameter containing the classes was missing, which meant this had to be created separately. This was done by creating a file, and adding the classes in the same order as the features; '1' for gripping and '0' for relaxing. Following this, the SVM was trained, producing the support vectors seen in the scatter plot in fig. 8. Further, K-Fold Cross-Validation with ten folds, which is also used in [6], was performed on the SVM model. This is done to estimate how well the model performs in practice, but can also be used to reduce sampling bias. Following this, the ten cross-validated models were analyzed on the test set. The model with the best accuracy was chosen, and used in real-time processing.

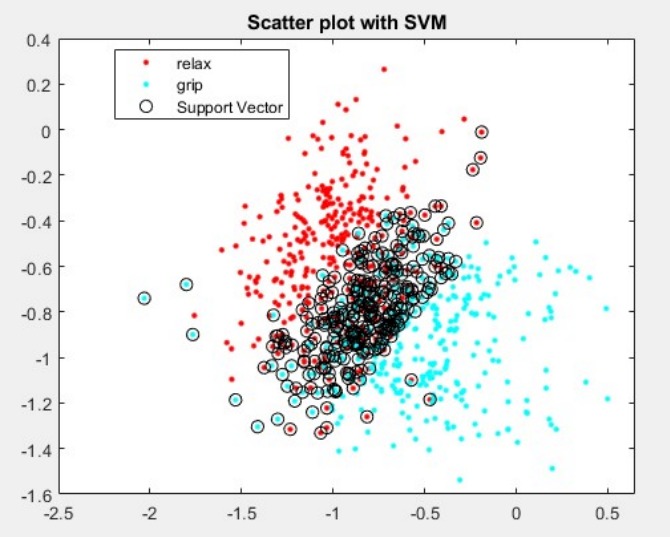


Figure 8. A scatter plot of the trained SVM model; showing the two different classes, relax and grip, but also showing the support vectors.

B. Component 2A - Real-time

1) Transferring data - Simon: The first part of component two is in reality also a part of component one, where the sample data is assembled into a window. When the window is full, it is then sent to component two. This is done in order for data not to be lost during the execution time of the processing and classification, done in the component two. This means that the program is free to process data while the window is being built in the component one.

The window creation is handled in LabVIEW through two while loops. The first loop receives data from MATLAB through UDP, saves the data in an array, and sends the full array to the second loop, as can be seen in fig. 9. After the array has been sent a new array is created and the loop continues, as seen in fig. **??**. In the second loop, the array is received and looped through in a nested loop. The nested loop is used to send the array to the next component over UDP, one sample at a time. This can be seen in fig. 10.

The samples are then received in MATLAB through UDP and assembled into a matrix, which is then used for the processing.

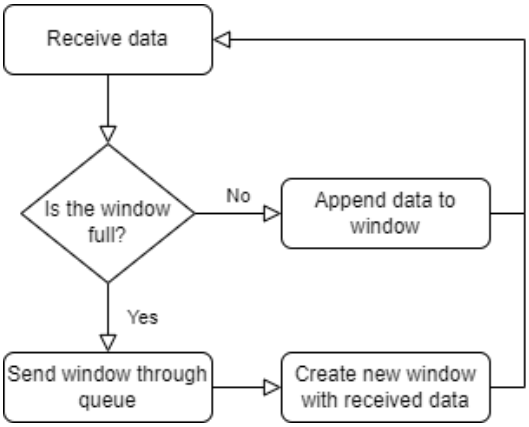


Figure 9. The while loop that receives data through UDP, and appends it to the window. When the window is filled it is sent through a queue to the loop in fig. 10, and a new window is created.

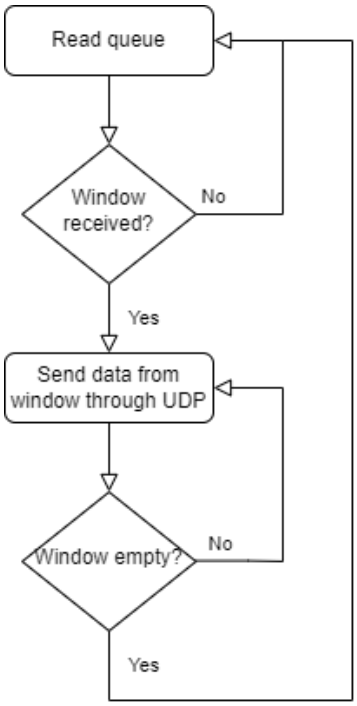


Figure 10. The while loop that is used to send the window one sample at a time over UDP. This happens when the window is received from the previous loop in fig. 9, through the queue.

2) Data analysis - Inas: Data analysis in real-time is quite different compared to offline, as the signals cannot be analyzed manually. Trials and channels which contain a lot of noise, which could potentially affect the further processes, cannot be dismissed in real-time. Due to this, channels which have shown a lot of noise prior in offline, have been permanently removed.

During progress of the project, component 1 had problems with packet loss when trying to send data. Although this was resolved later, data interpolation was implemented as a safety

mechanism, if packets is lost. This was done by testing three different approaches of interpolating data, and comparing the results to each other. First, every other sample was removed and then interpolated using the three different techniques. Secondly, the interpolated signals were first manually examined by comparing them to the original signal. This was done to get a hint of how well the techniques function. Further, cross-correlation was used to see how well the interpolated signals match with the original signal. The technique that matched the most with the original signal was chosen for the safety mechanism, see fig.11.

Before any kind of filtering and processing is done, the amount of samples received in the window from component 1 is checked. If the amount of samples received does not match the samples that is expected, in this case 100, it means samples have been lost. If this occurs, the ratio of the amount of samples that needs to be interpolated is calculated using the following formula:

$$F_{size} = \frac{Expected\_Samples}{Received\_Samples} \tag{3}$$

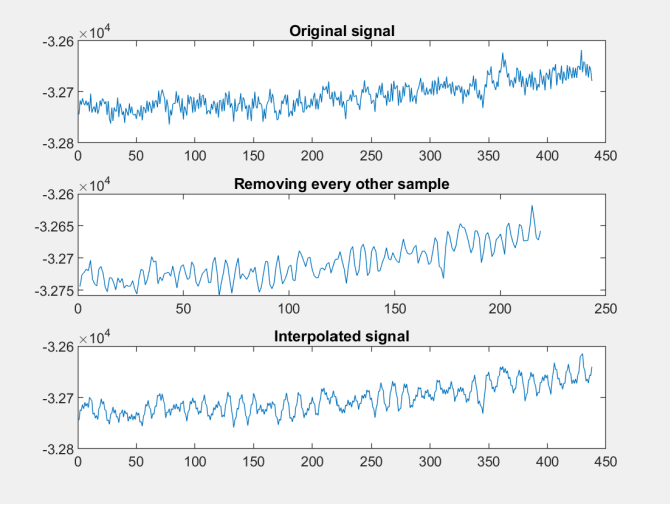


Figure 11. A subplot showing figures of the process of interpolating data; first figure showing the original signal, second showing the signal with removed samples, and the last showing the interpolated signal.

3) REAL-TIME PROCESSING PLEASE PUT TEXT HERE VICTOR:

4) Classifying and sending data in real-time - Simon: Using the extracted features from the processing, the window is then classified as being either "gripping" or "relaxing". This is done by using the trained SVM model and predicting the class. This class is then converted into the format used in component three, where "gripping" and "relaxing" is converted to "1000000" and "0000000" respectively. This is then sent to component three through UDP.

V. Method Component 1B

A. Software- Jonathan Holm

1) OpenBci: The sampling of the data from the electrodes was made possible through the OpenBci Gui. This software was made for the board on the EEG helmet, and was therefore selected for the acquisition part. The Openbci gui also provided multiple useful graphs; such as a real time head plot, where the user can see active electrodes and real time Fast Fourier Transform (FFT) plot. The program also provided an help option to check the impedance of the electrodes themselves, this to establish which electrodes had good connection against the scalp[45]. Which give beginners in the area of EEG a great start to collect quality data.

2) MATLAB: MATLAB was chosen as the program for collection of the data. MATLAB had a library to receive data trough LSL from the OpenBci Gui. When the data has been collected in MATLAB the signal can easily be manipulated and handled through MATLAB'S many intuitive matrix manipulations[46]. MATLAB was in addition selected as the data collection part of component 1B due to component 2B were too us the same program; this too minimize the amount of steps that the data had to travel to reach component 2B and therefore also minimizing the packets lost between the two components.

B. Data Acquisition- Pontus G

For data acquisition there are two defined states. Relaxed hand is one state, as the idle state and an closed hand is the active state. The states are only imaginary such that the test subject has to imagine an relaxed or closed hand in their head with no to little disturbances. The test conditions had a black screen with a white dot that could easily be focused on while performing the tasks. The protocol used were using a timer with six seconds interval, first six is imagining a relaxed hand and then a closed hand repeating this sequence ten times such two minutes elapsed per recorded session. For this project three different subjects were tested and only one subject were continued with doing a total of eight sessions(total of 16 minutes) during different days. The data gathered uses OpenBCI GUI [47] to read and stream data off the helmets 16-channel electrodes [34]. The sample frequency were 125Hz which were obtained through using the dongle that comes with the helmet. OpenBCI has the usability of saving offline data to text files after each session automatically for easy examination. Otherwise in real-time OpenBCI have good usability streaming data.

Protocol (one session)	
State	Time [sec]
Relaxed	6
Close	6
Repeat with same interval for 2minutes	

C. Connection OpenBCI/MATLAB- Pontus G

As mentioned in V-B OpenBCI has good streaming characteristics and especially for LSL and udp streaming. OpenBCI

simultaneously saves offline data to text file in the same time that it streams over via LSL and udp at a frequency of 125Hz over to MATLAB. This implies that the time for each sample can be calculated in 4. In 12 the real-time setup is described.

$$T = 1/f => 1/125 = 0.008 Seconds \quad (4)$$

.

Figure 12. (1) The helmet uses bluetooth communication with OpenBCI GUI. (2) The data is represented in OpenBCI GUI, impedance and connectivity are checked in this block before streaming. (3) via OpenBCI GUI the LSL and udp protocol stream data with 16-channels over to MATLAB.

D. MATLAB- Pontus G

1) LSL: To be able to read the LSL stream the LSL library needs to be loaded and configured. The program will search for an open LSL stream with the same type as being sent from OpenBCI (configured in OpenBCI GUI). When connection established between the two programs the text "Now receiving data..." is displayed and the stream from OpenBCI GUI can be started.

2) UDP: Udp is using a buffer which if it wasn't flushed it stored the same repeated data, such the udp socket isn't flushed automatically after read. Udp library were used in a MATLAB environment. From OpenBCI the local ip "127.0.0.1" is set and a configured port to stream and read from is set. In MATLAB the computer IP who is streaming is set and with same port number to be able to read and write from the same port. Also sending data over udp results in unwanted data in each sample such as the strings "type:EEG,DATA" which is removed [48]. Due it's one communication protocol it don't check for a connection just reads the port when data is available on it.

3) MATLAB: MATLAB then saves down each sample in an matrix of a total of 100 samples. The prediction and training of component 2B is done through packages of 100 samples per package which is why the matrix takes 100 samples before sending it to prediction. 100 samples represent 0.8s from 4. $TimePerSample \cdot Samples => 0.008 \cdot 100 = 0.8s$
Such one package takes 0.8s to acquire to MATLAB, for each package it is sent to a predict function from component 2B that is conducted through the same MATLAB script. The predict code then sends over the result through UDP to component 3, after the UDPsend is done MATLAB starts to gather a new package of 100 samples. This process is continued until the stream is closed.

VI. Method Component 2B

A. Preproccesing - Jonathan F

The preprocessing was made in MATLAB. The signal received from component 1 is changing between gripping and relaxing for 6 seconds each, throughout 2 minutes. The signal is then divided into 800ms windows with a size of 100 samples in each window. The first and last second of each action is removed due to noise and action-transfer (the movement between relaxing the hand to gripping).

Fast Fourier Transform (FFT) was performed to check magnitudes of the signals, see Fig. 13 A small noise at 50Hz appears, usually the peak is much larger but this has to do with the quality of the data.

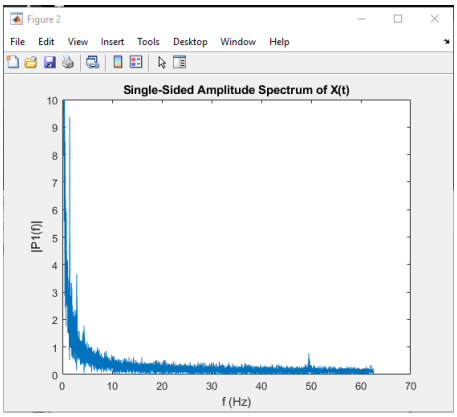


Figure 13. The frequency spectrum showing amplitudes of all frequencies between 0 and 62.5Hz

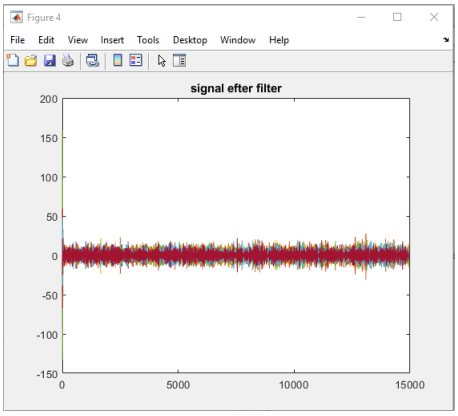


Figure 14. Signal after using bandpass-filter on all working channels. The signal is centered around 0.

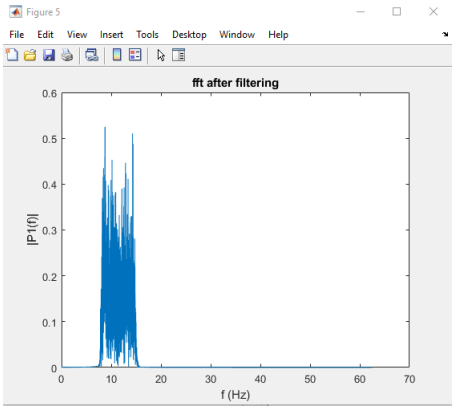


Figure 15. FFT after filtering signal, showing only the frequencies between 8-15Hz.

A few different filters were tested to find the best one suited for the project. By reading some papers about the same project it could be seen that a bandpass filter was mostly used. The filter were implemented and there were some peaks in the signal that was indescribable at the time. When the signal was plotted in time, it did not match the gripping and relaxing windows. First i was later discovered that a notch filter was not needed because the bandpass is removing the peak. Both FIR-filter and IIR-filter for the bandpass were tested. However, a good and suitable FIR-filter were never really found even though that was mostly used in previous work. So, the bandpass IIR-filter was finally used with a filter order of 30 and frequencies between 8-15 Hz. In fig 9 only the frequencies from 8-15Hz can be seen.

B. Proccesing - Karl

After preprocessing, features that can differentiate between a closed and a relaxed hand need to be extracted. The chosen feature extraction method is Common Spatial Patterns (CSP). CSP was introduced by H. Ramoser to be used in classification of multi channel EEG during imagined hand movements [49]. It works by maximizing differences in variance between two classes, in this case the two classes are closed and relaxed hand. The algorithm used in this report is described as follows [50]. $\mathbf{X}_o$ and $\mathbf{X}_c$ denotes the filtered EEG signal obtained of both classes (closed and relaxed) from the head cap in matrix form. Its dimensions are [N x C x T] where N is the amount of samples in one trial, C is the number of channels and T is the total amount of trials.

$$C_o = \frac{X_o X_o^T}{trace(X_o X_o^T)} \qquad C_c = \frac{X_c X_c^T}{trace(X_c X_c^T)} \tag{5}$$

In matlab the cov function accomplish this [51]. $\mathbf{X}^{\mathrm{T}}$ is the transpose of $\mathbf{X}$ and trace($\mathbf{X}$) is the computes the sum of diagonal elements of $\mathbf{X}$. $\mathbf{C}_o$ and $\mathbf{C}_C$ are then computed for all trials to produce T amount of covariance matrix for each class, the matrices has the size [C x C]. The covariance matrices are then averaged over trials to produce $\mathbf{M}_o$ and $\mathbf{M}_c$. The composite spatial covariance matrix can be described as.

$$M = M_o + M_c = USV \tag{6}$$

Where $\mathbf{U}$ is the matrix of eigenvectors and $\mathbf{S}$ is diagonal matrix of eigenvalues. To obtain $\mathbf{U}$, $\mathbf{S}$ and $\mathbf{V}$, Singular Value Decomposition was applied, in matlab the svd function can be used [52]. The whitening matrix is calculated with this formula.

$$P = S^{-\frac{1}{2}}U \tag{7}$$

$\mathbf{P}$ can the be used to transform the averaged covariance matrices $\mathbf{M}_o$ and $\mathbf{M}_c$ into:

$$A_o = P^T M_o P \qquad A_c = P^T M_c P \tag{8}$$

$\mathbf{A}_o$ and $\mathbf{A}_c$ share common eigenvectors and the sum of their diagonal eigenvalue matrices will always be one.

$$A_o = JG_oH \qquad A_c = JG_cH \qquad J_o + J_c = I \tag{9}$$

Singular Value Decomposition was used again to extract $\mathbf{F}$, it will be used to construct the spatial filter matrix. Note that eq. 5 will produce the required $\mathbf{F}$ regardless of whether $\mathbf{A}_o$ or $\mathbf{A}_c$ is used because of similarities. This can save time as only one needs to computed. $\mathbf{W}$ is denoted as.

$$W = JP \tag{10}$$

The columns of $\mathbf{W}$ are the spatial filters, which can transform the original EEG signal into uncorrelated components. Where $\mathbf{F}$ is the filtered EEG signal and $\mathbf{Z}$ is the EEG source components. Another easier way to obtain a similar W is to use the eig function in matlab [53]. In this report the Matlab eig function was used due to easier implementation and very similar result as previous stated equations.

$$Z = WF \tag{11}$$

The features to be used in classification is the natural logarithmic variance (logvar) of channels in one trial. A comparison can be seen in fig. 16 and fig. 17 where the first figure is the logvar before CSP was applied and the second is the logvar after.

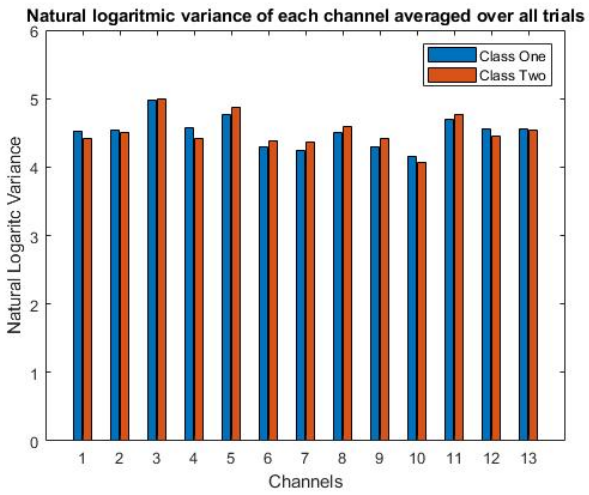


Figure 16. This bar plot shows the averaged natural logarithmic variance of each channel before CSP was applied, the blue bar is the relaxed hand class and the red bar is the closed hand class.

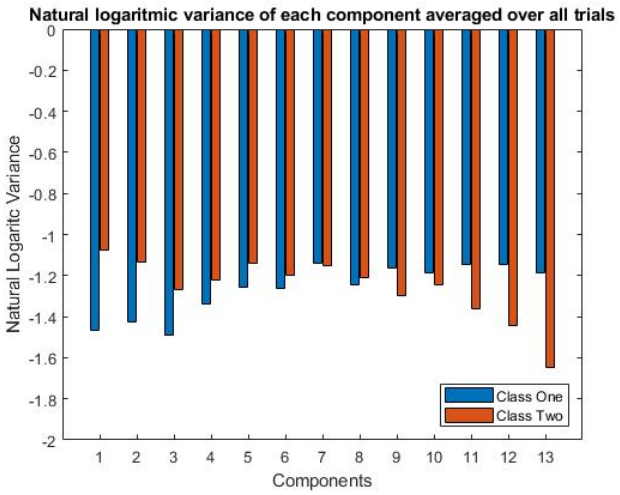


Figure 17. This bar plot shows the averaged natural logarithmic variance of each component after CSP was applied, the blue bar is the relaxed hand class and the red bar is the closed hand class.

As input to the classifier both the first and last component of each class will be used, this is because they have the largest difference between the classes.

C. Classification - Karl

Once the Features from the processing stage are extracted, to decide between whether the hand is closed or relaxed a classifier is needed. Support Vector Machine with a linear kernel is the classifier of choice. SVM is a robust classification and regression algorithm with multiple fields of application [54]. The data was split into a 80% training set and a 20% testing set. It was ensured that both the training and the testing set contains equal amounts of both classes, this is to have a fair representation during training an especial during testing. The split resulted in 640 trails in the training set and 160 trials in the testing set. A label vector was also created, where a '0' represents class one and a '1' represents class two.

In order to improve performance of the SVM, an optimization was done. Matlab has an inbuilt optimizer for SVM which optimizes the hyperparameters of the SVM to minimize k-fold cross-validation loss [55]. The model was tested on the testing set to ensure sufficient performance and will then be used in the real-time part.

D. Online - Karl

The real-time part contains preprocessing, processing and classification all in one. Once one windows is obtain from the head cap via the LSL stream, the signal is filtered with the previously mentioned band-pass filter, then the signal is multiplied with the spatial filter matrix $\mathbf{W}$, see eq. 7. The first and last component from the result of eq. 7 is then extracted and fed to the SVM model, which will make an prediction. When the prediction is done it will be sent to component 3 via UDP and the result will be displayed in form an relaxed or closed robotic hand.

VII. Method - Component 3

(Emanuel and Joakim)

First, the existing components were examined. There was a roboRIO with cables and installation DVDs, two arms with hands and some related components, three special servos and two continuous servos. Both arms were cracked and only had mounting options for one servo, and not five. The fishing line used to pull the fingers was cut off. The springs used for tensioning the fishing lines were too long, and cut in a way that was both unpractical and bad looking. The continuous servos could not be used because they cannot hold an angle, and the special servos could not be used since they used an uncommon communication protocol that has only been implemented for Arduino UNO and is poorly documented. It was therefore decided that making a new library for the roboRIO is outside the scope of this project.

Therefore, it was decided that all the hardware except for the roboRIO and the hands should be replaced.

A. Setting up LabVIEW for roboRIO

(Both)
LabVIEW was not possible to install from the roboRIO DVDs since no computer had a DVD reader. It was therefore decided that the online download options from NIs website should be used.
(Emanuel)
First LabVIEW 2021 64-bit was installed along with drivers, it did not work. Further testing seemed to show that the roboRIO does not work along a 64-bit versions of LabVIEW. Therefore, LabVIEW 2021 32-bit was installed with the 32-bit versions of all drivers, it did not work either. Further reading online mostly described a software called FIRST. The software called FIRST only worked with Visual Studio Code, and not LabVIEW. Further reading into the problem seemed to show that the roboRIO would only work with older versions of LabVIEW 32-bit if roboRIO toolkit is installed.
(Both)
Every edition of LabVIEW 2020 and 2019 was tested with no luck and it was not possible to download roboRIO toolkit from NIs website.

(Joakim)
In order to get an understanding of how the RoboRIO would connect to LabVIEW, one needed to look deeper into the RoboRIO system specifications (this was done on a Windows machine). To simply use LabVIEW built-in functions did not work. There was nothing wrong with the RoboRIO device itself, as the test panel application that comes with it indicated that it was connected to the computer, and that its outputs worked fine (after several downloads and modifications to the system). In National Instruments (the company that created RoboRIO) application "NI Measurement and Automation Explorer", and also by typing its IP adress into a browser, one could investigate the properties of the RoboRIO device. It had support for two versions, the 2016 and 2019 version. The 2019 version was already installed in one of the student's computer, so this was tried with. However, this did not work as intended, because this version was missing the RoboRIO/FRC kit necessary to implement functions that enables the user to connect and control the device. These kits were available online, but only in 2015 and 2016 versions.

(Emanuel) After some further research, it seemed like the roboRIO will only work with LabVIEW 2016 32-bit alongside roboRIO toolkit. All previous software was uninstalled and LabVIEW 2016 32-bit was downloaded, but it needed a license that the school did not have, and would not complete the install.
(Both)
At this point, it was decided that the only viable option was to get hold of a DVD reader and install the software and drivers from the provided DVDs. After a slow install from the DVDs, the software could communicate with the roboRIO and generate PWM signals on the PWM ports of the roboRIO.

B. Software

Figure 18 shows how component 3 process a command, from received command send by component 2, to a PWM signal output on the roboRIO. The following sections explain each software component in further detail.

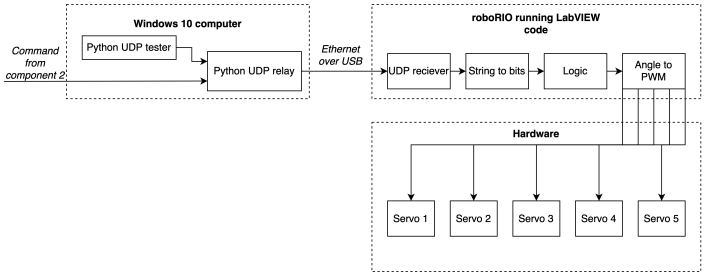


Figure 18. Flow diagram of the logical layout of component 3. Control commands are sent from component 2, or generated by a local Python script. The commands are then relayed from the Windows computer to the roboRIO using LabVIEW to do all processing in order to make the servos turn as commanded.

The message was encoded as an UTF-8 string containing seven digits that could be a one or a zero. The two first digits sets the state of the system: 00 - hand open, 01 - hand half open, 10 hand closed, 11 finger control. If the first two digits were 11, then the system reads the following five digits, one for each finger, where a 1 shows finger closed, and a 0 finger open. In any state that was not 11, the last five digits were ignored.

1) UDP Relay - Emanuel: To transfer data from component 2, it was decided to use IP/UDP. Because the roboRIO is on a separate network than component 2, a Python script was written that creates an UDP socket and receives messages, and relay them to the roboRIO through Ethernet over USB. Each message is also written to the Windows Command Prompt to ease debugging.

2) UDP Tester - Emanuel: In order to test component 3 without needing component 2, a Python script was written that sends different commands to the IP/UDP socket of the UDP Relay. This simulates all the different commands that in theory could be sent by component 2. Open, half open and closing of the entire hand, and open and closing of each individual finger was sent, one after another in an infinite loop to test that component 3 was working according to specifications.

3) UDP Receiver - Emanuel: An UDP socket was created in LabVIEW on the roboRIO that listens for messages from the UDP Relay. Each message is received as a string of ones and zeros.

4) String to Bits - Emanuel: Since each digit in the received string represented different commands, the seven-digit string had to be split into seven individual numbers to make them easier to work with. First the string was converted into a double, then the formula $\frac{X}{10^n}\%10$ was used, where X is the input string, n is the index of the digit to extract and $\%$ represents modulus operation. For example the equation for the first digit would be $\frac{X}{10^6}\%10$ and for the second digit: $\frac{X}{10^5}\%10$.

5) Main program template - Joakim: This LabVIEW Virtual Instrument file demonstrates how the whole system from component three works, software wise. From receiving the 49 different input combinations (7-bit) from component two, to displaying angles corresponding to these for each finger. At first, this file had the purpose of displaying and simplifying the 7 bit input and 5 finger output to make debugging easier. Later it also got the purpose of demonstrating the system for a first time user. A description of the system, step by step, can be found below.

- Handling input bits to commands

There are 7 buttons representing each bit. The first two are compared with logic OR, AND and NOT gates, as the table in results section "Main program and its template" shows the result of. The four alternatives will display a light indicator if used. Only one of these "Test" indicators can be TRUE/ON at the same time, determined by the gates. This shows which one of the first four states is available for use.

- Commands and what they do

The three first alternatives are made of sequence blocks that sends their commands further to a case structure, if activated by the two first buttons. The fourth alternative is a while loop that enables individual finger control buttons as long as the first two buttons are on. Otherwise, the finger control will not change, despite pressing their five buttons. The individual fingers have their own indicators. It begins with "Finger 1" or thumb, and ends with "Finger 5" or little finger. Each finger sends its signal further to its case structure, similar to the three first states.

If there is a signal inside one of the three first case blocks, that one sends its approximate degrees of flexion further (in the main program these are then adjusted in "Angle to PWM" so that fingers do not collide with thumb or palm). In the other case blocks, this signal does not get sent. The five finger case blocks can concurrently send either full flexion or no flexion, if the while loop is activated. If that is the case, a TRUE or FALSE signal will always reach the five cases. These are determined by the five finger control buttons.

- Handling commands to output

Commands from selected case blocks are then converted to an appropriate data type and displayed on gauges in real time. The gauges display no flexion or "Open"/"Relaxed" (0), in between flexion or "Between open and closed" (90) and full flexion or "Closed" (180). By changing the case blocks, the gauges will display other values than the ones chosen.

6) Main program - Joakim: This is an extension of the template. It includes software for receiving messages via UDP (instead of using buttons), converting these messages to bit form instead of buttons, and sending a message to the RoboRIO device to control servo positioning using PWM. All in real time.

More thorough instructions on how to use the main program can be found in the "Getting started guide" for component 3. It was made after finishing the files that controlled the system. The purpose of it is to make it clearer for new users how these programs behave, and thus letting them continue to build upon and optimize it.

7) Angle to PWM - Emanuel: Because the code was using angles in degrees and the servos need a PWM signal with varying duty cycle to represent an angle, one sub VI for each finger converted the angles in degrees into duty cycles and then generated the PWM signals. Two servos were mounted with the wheel to the right and three servos were mounted with the wheel to the left. Because of this, two servos were rotating in the opposite direction from the other three. The servos were assumed to be linear, therefore only two points were measured: 0 degrees and 180 degrees. Then two linear equations were calculated, one for each servo orientation. The equations took an input angle in degrees and output the equivalent duty cycle. The duty cycle was then fed into the PWM VIs included with roboRIO toolkit to generate one 50 Hz PWM signal for each finger with the calculated duty cycle.

C. Hardware

1) roboRIO: The roboRIO was connected to the Windows 10 computer using an USB cable. The roboRIO also had an external power supply to provide power, it also had to be connected.

2) Servos - Emanuel: Because of the ongoing pandemic, many products are in short supply, especially electronics. The Hitec HS-422 [56] servos was chosen because they were in stock, was cheap, decent speed and strength compared to other similar servos, could be run off the 5-6V provided by the roboRIO and uses a PWM signal as input to represent an angle.

3) Servos - Joakim: Before receiving the Hitec servos, a plan B was made. The continuous Parallax servos were not able to stay in a position with high resistance (for gripping). After some regulations to the PWM signal frequency and duty cycle [57], they were able to achieve this. They could also get to an angle by approximating how long time it would take to move between the current angle to the desired angle at some known speed. Therefore they could have been an alternative to the Hitec servos, but angle drift would probably have been a problem over time. Thus, the Parallax servos were not used as the Hitec servos are superior when it comes to holding a known angle, especially under load.

4) Forearm - Emanuel: The requirements of the forearm was to fit the already excising hand, hold five servos, provide decent alignment such that a string could be run to each finger without tangling, be as small as possible and be easy to 3D print.

First one photo of the mounting point was taken using an iPhone 12 Pro. The camera was located far away from the mount and zoomed in as far as possible to reduce the effects of perspective distortion. To align the camera straight in front of the mount, and make sure the camera was not looking at the hand from an angle, the flashlight of the iPhone 12 Pro was turned on, such that the light was reflected off the flat mount. When the reflection was as strong as possible, and therefore the alignment very good, the photo was taken.

The photo was imported into SOLIDWORKS and placed on a plane where the outline of the mount was sketched using splines, and later extruded into a 3mm thick solid. The solid was 3D printed to make sure it fit the real hand, and it did. That meant that the sketch of the mount was good enough in both skew and scale.

Models of the servos and the straight arm were downloaded from GRABCAD[58] and then imported and saved as two separate parts.

To measure how far the servos has to pull each string, each string was measured when the fingers were in open-and-closed position, and the difference in the measurements was how far the string has to be pulled.

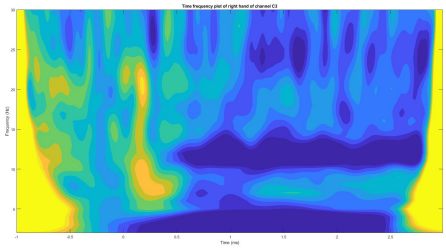
The wheels that fit the straight arm of the servos were designed and 3D printed. After mounting a wheel to a servo, it was strong in the direction parallel to the arm but could wiggle too much in the direction perpendicular to the arm. Multiple models of the X shaped arm were downloaded but all of them had incorrect measurements. A X shaped arm was designed using the photo technique mentioned earlier, to provide as a template for designing a new wheel that fits the X arm. The new wheel design was 3D printed and mounted to the servo and it was very stiff in all directions.

Then an assembly was created in SOLIDWORKS where the mount of the hand, and five servos with X arms and wheels, were imported and mated. Everything was aligned in a manner that the strings could be routed straight from the wheels to the fingers, next to each other whilst minimizing the space used. Then a big solid was designed that had mounts for all five servos and merged the servo mounts with the mount of the hand smoothly. A link to a GitHub repository was also added to the bottom of the forearm to make it easy for future students to access all design files and code needed to use component 3. Then the forearm was 3D printed. It took 12.5h to print with 0.35mm layer height and a 0.6mm nozzle.

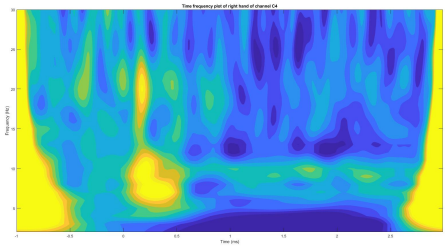
Finally, all components were mounted onto the forearm with zip ties. New strong and agile strings were used along with smaller springs designed for pulling, to connect the wheels to the fingers. When everything had been assembled, the angles of the wheels, for when the fingers are: open (wire was tense but not pulling), half open and closed was measured and the variables in the logic were changed to reflect the real world measurements.

5) Testing component 3 - Emanuel: The entire system of component 3 was connected as planned and tested by sending messages using the Python UDP tester to make sure everything works properly, both the software and hardware together. To measure the delay from received package, to when the hand moves, a camera recording at 24 frames per second filmed the computer screen and hand. By visually counting each frame, from when the message is printed on the screen by the UDP Relay, to when the hand moves, an estimate of the delay of component 3 could be calculated by multiplying the number of frames taken with $\frac{1}{24}$ seconds.

6) Testing with component 1b and 2b: Group 1b, 2b and 3 met to test the entire system, from brain to hand, and after disabling Windows Firewall and correcting the IP addresses used, the system could use brain signals to control the hand.



(a) The time frequency plot of the channel 'C3' of IM right hand movement data.



(b) The time frequency plot of the channel 'C4' of IM right hand movement data.

Figure 19. The time frequency plot of 2 channels of IM right hand movement data. The time frequency is calculated using Complex Morlet Wavelet with 8 cycles frequencies between 2Hz and 30Hz. The 0 time in the time axis represent the on cue time to perform the task.

VIII. Result Component 1A - Ludwig W

The offline solution produced 15 sessions where data from 13 functional electrodes where used. The data from these 15 sessions where saved and used to validate the package loss for the system. The validation of the package loss was done through comparing a session saved to a .txt file from OpenBCI to the same data being saved after being transmitted throughout the system. After comparing more than 1000 samples, all samples was identical, thus had been correctly transmitted to component 2A.

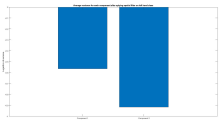
The data contains no labels that describe the subjects intent nor does the signal contain timestamps which prohibits the validation of the whole systems real time constrains.

IX. Result Component 2A

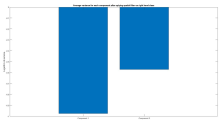
A. victor

The plot of the time frequency generated from the wavelet analysis is shown in figure 21 for channels 'C3' and 'C4' of the right hand movement class. After applying a spatial filter , the feature extracted i.e logarithm of the variance of the data is average over all epochs and the result can be seen in figure 20

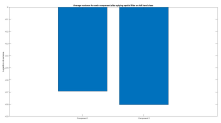
The scatter of the variance of right hand movement class and left hand movement class after applying the spatial filter is shown in figure **??**



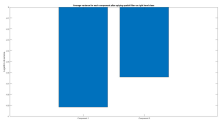
(a) The average of the variance in the spatial filter component directions for Left hand movement class of training data



(b) The average of the variance in the spatial filter component directions for right hand movement class of training data

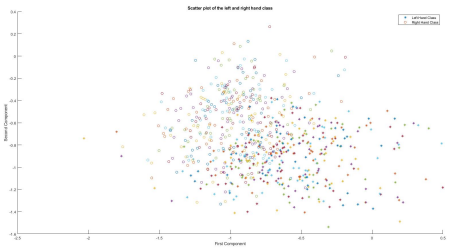


(c) The average of the variance in the spatial filter component directions for Left hand movement class of test data

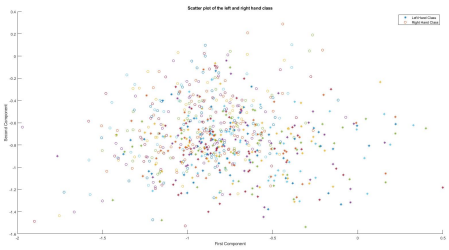


(d) The average of the variance in the spatial filter component directions for right hand movement class of test data

Figure 20. The average of the variance of the data after applying the spatial filter on test data and training data.



(a) The scatter of the variance of both classes of the training data.



(b) The scatter of the variance of both classes of the test data.

Figure 21. The scatter of the variance of the data after applying the spatial filter. Each point on the scatter correspond to one classification input

B. Offline

The SVM model was successfully trained with 720 samples, and validated using K-Fold Cross-validation with 480 samples. The best model, out of ten, had a accuracy of 55%.

C. Real time - Simon

The real time part of component 2A worked as planned. Through some cooperation with component 1A the data stream was assembled into windows in LabVIEW, which were then sent to MATLAB. In MATLAB the data was successfully processed and classified, with the classification sent to component 3 through UDP.

X. Result Component 1B - Jonathan H

The results from trying two different protocols from OpenBci to MATLAB;
As can be seen in 22 the amount of data point from the UDP is sufficiently lower than then actual data. The ratio between captured and lost packets was 5 to 20, the latter being the lost ones.

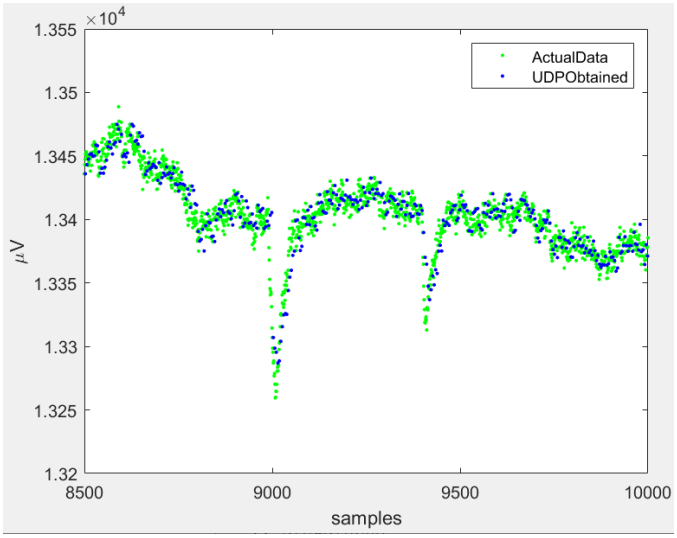


Figure 22. A picture of the results given by using the UDP protocol. The green point represents the data collected and saved by the OpenBci Gui, and is labeled as the actual data. Blue point being the data collected in MATLAB using the UDP protocol from OpenBci Gui to MATLAB.

The use of LSL gave a higher amount of samples in MATLAB. This can be seen in 23, 3 samples was missing from the start of the session out of 15000 total samples.

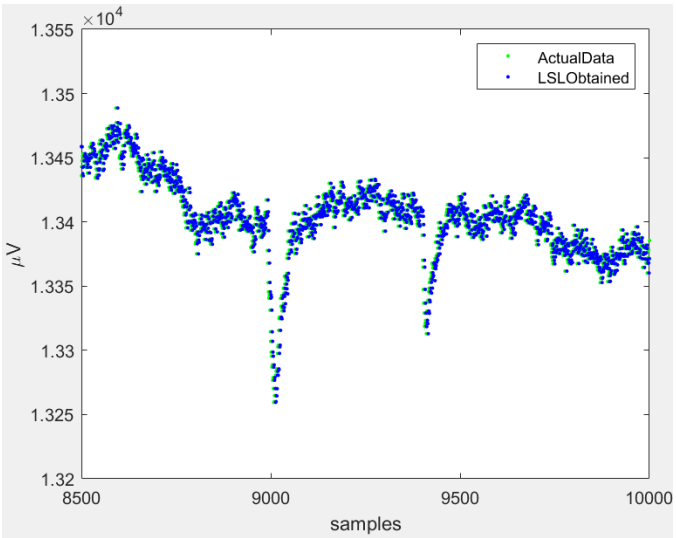


Figure 23. A picture representing the data collected in MATLAB when data was streamed from OpenBci via LSL. The green point being the actual samples saved by OpenBci offline. Blue is the LSL data. The graphs are almost identical, the fluctuations in the signal is captured even in MATLAB but is shifted 3 samples back.

XI. Result Component 2B - Karl

The CSP is able to separate the channels/components. This can be seen in fig. 16 and fig. 17, where the classes differ from each other more in the latter figure. The accuracy from the SVM model on the test data was appr. 60%. A scatter plot of the model and its support vectors can be seen in fig. 24.

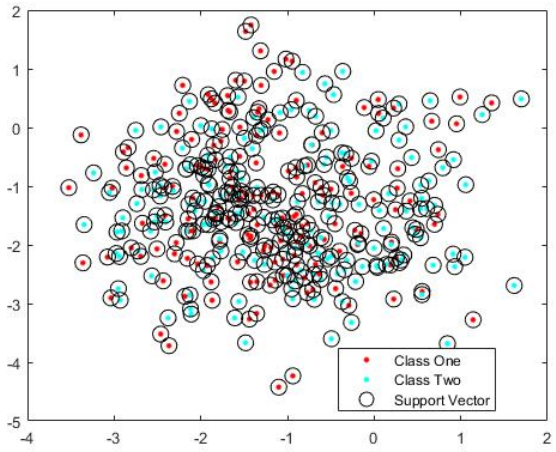


Figure 24. A scatter plot of the SVM model generated from the hyper parameter optimization showing both classes and the support vectors

XII. Results - Component 3

(Emanuel and Joakim)

Overall, all parts of component 3 worked as planned. There were a few issues along the way, but everything could be resolved to produce a working component according to specifications. All files can be found on GitHub at `github.com/Emanuel-Bjurhager/robot_hand/`

A. Setting up LabVIEW for roboRIO

After wasting a lot of time, the only working software was that from the DVDs that came with the roboRIO.

B. Software

The software worked as expected. A few minor issues were encountered, but all could be resolved somewhat easily.

1) UDP Relay - Emanuel: It was not planned that this part would be needed, but after discovering that the computer and roboRIO were on different networks, it seemed easiest to write a relay using Python, and it worked on the first try.

2) UDP Tester - Emanuel: This code worked flawlessly, with no issues.

3) UDP Receiver - Emanuel: Because of inexperience with LabVIEW, and Windows Firewall sometimes blocking communication, there were some issues with receiving data, but after figuring out how LabVIEW works and disabling Windows Firewall, there were no other issues.

4) String to Bits - Emanuel: There were no big difficulties encountered. The code worked as intended.

5) Main program and its template - Joakim: The main file (Main.vi) connects to other files and RoboRIO correctly using the RoboRIO LabVIEW 2016 DVD with both Windows and Mac OS. The template file demonstrates the main file function as intended. Both files consists of the following robotic hand states:

Command (bits)	Event
All finger movement	
00*****, 1100000	'Open' (relaxed)
01*****	Between 'open' and closed
10*****, 1111111	Closed
Individual finger movement	
1100001	Only little finger closed
1100010	Only ring finger closed
1100100	Only middle finger closed
1101000	Only pointy finger closed
1110000	Only thumb closed

Where '*' in the table shows that any change to this bit will not induce any change to the system. Individual finger movements can be combined, for example command '1110001' would close thumb and little finger simultaneously (with some modification to avoid collision). By regulating "Angle to PWM" files, any finger can take any state that the 3D-printed hand and servos allows for.

A "Getting started guide", for a user to learn what the component 3 program does in more detail, can be found as a file in Neurotechnology 2021/22 Teams group, or by contacting Joakim Nylander Nordström[‡].

6) Angle to PWM - Emanuel: This worked as intended.

C. Hardware

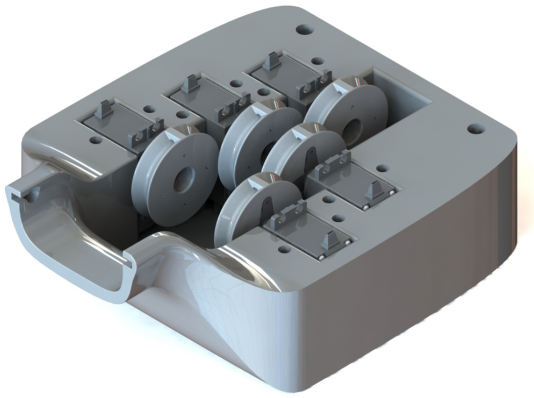
The hardware worked as intended in the end, but the forearm has some issues with a layer cracking that could be fixed with glue. The servos were not being held good enough with a zip tie, but after inserting some thick double adhesive tape between the servo and zip tie to make the servo thicker, it was held securely in place.

1) roboRIO: The roboRIO functioned correctly once all drivers and software had been installed.

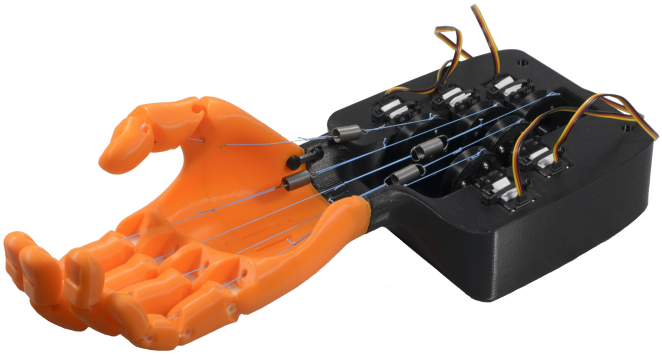
2) Servos - Emanuel: The servos worked as intended and were faster than expected. They were easy to work with.

3) Forearm - Emanuel: There were some difficulties with SOLIDWORKS trying to make an inverted volume and crashing when trying to merge the servos mounts to the mount of the hand. Guide curves were used to help SOLIDWORKS figure out how to create the solid. There was an issue with the 3D print where the holes for the servo screws should have been 1.5 mm but turned out smaller because of plastic shrinkage around the holes. A drill was used to get the holes into the correct diameter. Gravity also caused two circular cutouts around the hand mount to become slightly oval. When using a knife to expand the hole, a crack formed between two layers. Glue was used to keep the layers together. It was very difficult to tie the strings at the correct length. A difference of a few millimeters could mean that it was too stiff or had too much slack. After many hours, a good way of tying knots was discovered that

helped with placing a knot at a very precise location on the string.

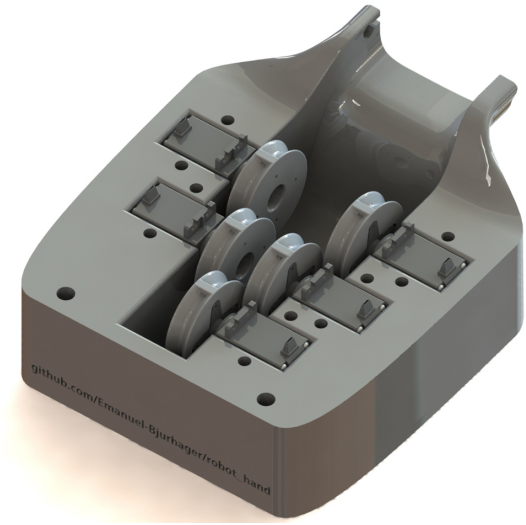


(a) Rendering of forearm assembly in SOLIDWORKS

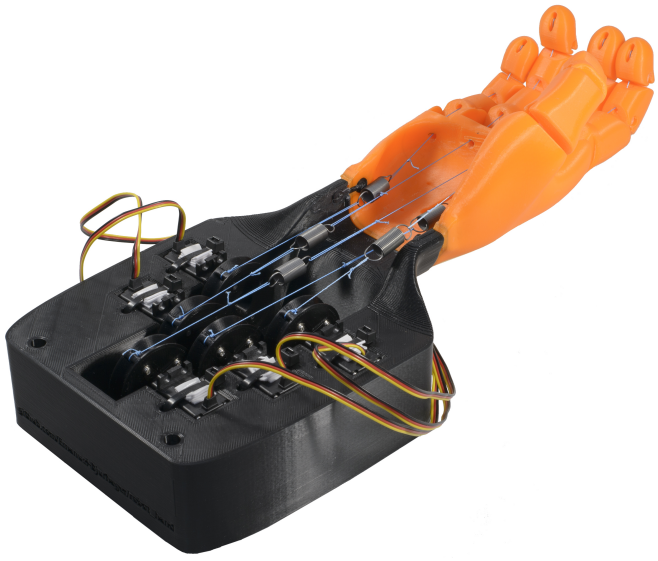


(b) Picture of hand

Figure 26.

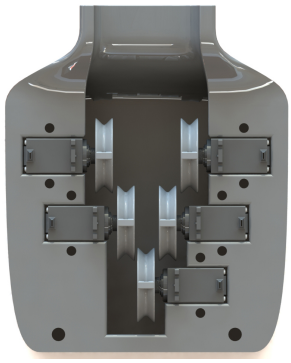


(a) Rendering of forearm assembly in SOLIDWORKS



(b) Picture of hand

Figure 25.



(a) Rendering of forearm assembly in SOLIDWORKS

(b) Picture of hand

Figure 27.

4) Testing component 3 - Emanuel: The test revealed a bug that sometimes caused two fingers to not move in some situations; it was easily resolved by correcting an error in the LabVIEW code. The delay was about 2 frames, therefore the delay of component 3, from received message to physical movement of the hand was about $\frac{2}{24} = 0.083$ seconds. Since the UDP Tester sent all different commands, and all of them worked. It could therefore be concluded that component 3 fulfills everything that was expected from it.

5) Testing with component 1b and 2b: After using the correct IP addresses and disabling Windows firewall on all computers, the system seemed to work as intended. Component 3 correctly read all messages sent by component 2b and moved the hand accordingly.

XIII. Discussion Component 1A

A. Fredrik

During the recording sessions we were extremely careful and consistent in detaching the cords (to avoid the cords from twisting off) from the electrodes before twisting the electrodes in to place. Still, it was discovered along the project that several wires were broken. The cause of this, and when it may have occurred, is unknown. They may have been broken before we initiated our experimentation. To produce as good data as we could, we screwed the electrodes until the OpenBCI GUI showed an impedance < 600 $k\Omega$ for each channel correlating to an electrode. The broken electrodes/wires were discovered as some channels constantly produced noisy signals or never went below the impedance threshold. No multi-meter was available to test the electrodes or cords before strapping on the headset.

In addition, the Bluetooth and wifi connectivity was unreliable at times. Some sessions took up to 30-40 attempts before a connection was established. Sometimes restarting the cyton board and/or restarting OpenBCI seemed to help, but not always. When investigating this further we found a disclaimer stated on OpenBCIs own website that the equipment did not fulfill the european CE standards. We believe therefor that our connectivity issues may have been a result of electromagnetic interference.

In addition to this, being strapped to the Ultracortex headset while waiting for Bluetooth connection and for all channels to have acceptable impedance, was extremely fatiguing and painful (both physically but also mentally). It is unknown to us how much effect the fatigue and pain actually had on the signal quality and data produced, that component 2A had to use.

The issues listed above gave us no time to add a user interface with instructions to the user, as the focus was to have actual data to send to component 2A. So we retrieved a black sound absorbing folding screen, attached a white dot for the user to focus on, turned off the lights and closed off the room from other people to ensure that limited external factors would interfere with the signal in the recording.

XIV. Discussion Component 2A

A. Victor

14 person working on the same project needed more structure that what was done in this paper. The component was working in small groups and there was not good communication between the components and the groups. It is helpful for the whole project to create a good structure from the beginning of the project as it will result in better report and project as a whole.

The data of left and right hand IM movements does not have the same neural activity of gripping and relaxing IM hand movements as the right - left will create a suppression in the μ band on different sides of the brain. While the data that should be used in this paper does only create a suppression on one side of the brain. This means that the algorithm still need to be tried on the relaxing and gripping data to conclude it's effect in this case.

The CSP algorithm used in this paper is working as expected but it is over fitting the data. Different size of training and testing set may be used to get better results.There is multiple CSP algorithms a particular algorithm that may be helpful for this project is to use the data from the μ frequency as first class and data that does not contain μ frequency as the second class. The algorithm will find direction in the data that maximize the difference between the classes. When applied on the data the μ frequency will be intensified in both classes but as there will be suppression in the gripping class the relaxing class will have a higher power at the μ rhythm frequency.

B. Inas

When testing the SVM model with K-fold Cross-Validation, it gave a result of 55% accuracy. Although this number is low, and perhaps not what was expected, there are some improvements which could result in higher accuracy. Instead of splitting the training and data set 60% respectively 40%, one could try different splits; such as 80% and 20% or 90% and 10%. By increasing the training percentage, there is more data to train on which means the model has a better chance of arriving at the correct answer. Even though this improvement could be quick and simple to implement, due to time constraints this was not a available option. Another method which could improve the accuracy, is to attempt to use different validation algorithms to validate the SVM model. Although there is a low percentage of chance to overfit with SVM and K-fold cross-validation, one could try to validate the data using Nested Cross-Validation to reduce the risk of overfitting. By using different validation methods, one could compare and pick the one with best results. Another improvement could be to add more data, as this would reduce assumptions and weak correlations, which would result in a more accurate and better model.

XV. Discussion Component 1B - Jonathan H

When the project started a lot of time was misspent; as the equipment had some troubles working from the start of the project. Different methods was tired on 5 computers of

varying brands. Yet no connection was established between a computer and the EGG helmet; and the rare times it got a connection it could take up to 20 minutes to get a stable one.

When a solution for the connection error was made, another problem was confirmed. Being able to get the data from the OpenBci Gui created some problem and many of the early weeks was only debugging methods that was thought too work in the first place. When a first successfully way was established through UDP, a lot of problem was discovered with this protocol. The UDP buffer only contained 5 identical samples that repeated until the stream was terminated. Making the signal recorded not contain any information of significance. The debugging then continued and after some trial and error, it was discovered that flushing the UDP buffer in MATLABA increased the amount of collected packets. Unfortunately the flushing came with it's own problems; when the flushing was initiated 20 samples was lost. Meaning that lost amount was 4 times the captured ones. And within component 1B it was discussed if this was too small of a sample amount, so a different approached had too be the answer. LSL was then tried, and with this protocol only 3 packets was lost under a 2 minute recording session with 15000 packets in total. The electrodes on the helmet itself, is brittle, and the cable inside the electrodes could easily be twisted of with a simple mistake of leaving the cable connected when being turned. The project started with 16 working channels but ended up with 13 at the end. This resulted in that less information was recorded under the later parts of the project. This is not taking into consideration the potential that maybe some of the cables might be partly damaged. Damage not visual by eye.

When it comes to future work regarding the acquisition part of the project;

- As some unexpected things happened under the start of this project that caused less time too be available too other things then finding a way to transfer data from OpenBci Gui. Some improvements could be to try to make a visual protocol rather then using an auditory one, which was used in this project due to time constraint. By eliminating the sound itself one can maybe improve the overall signal being recorded.
- The project could also maybe improve if, more data from more people was given to the prediction part of this project (component 2B).

XVI. Discussion Component 2B

A. Jonathan F:

The aim of the project was to develop a complete real-time neurorobotic system to open and close a robotic arm, using the mind. This was done correctly and the hand moves (relaxing and closing) when thinking about it.

The signal was hard to understand sometimes. FFT was used to analyze the signals but by reading more papers later on, it could be seen that Power Spectral Density (PSD) would be better, since it measures the signals power content versus frequency.

Features were extracted using CSP. CSP was chosen because of its benefits in EEG-based BCI systems for MI. The EEG signals have noise and over-the-fitting-issues, which will be taken care of by using CSP. The results after the implementation of CSP were found better than when CSP was not implemented in the system. This proves that a good spatial filter is needed in signal processing, to reduce spatial noise that cannot be handled by standard time-frequency filter.

There are some alternatives for CSP that could be tested in future work to provide even better results. Independent Component Analysis (ICA) can be used, it is preferable when the experimental condition is novel, or labels are not available.

SVM is used because it is the best MI technique used for classifying EEG signals, especially mental tasks used in this project, like thinking to move the hand. EEG signals are represented into high dimensional feature space for analyzing the brain activity, where SVM have shown good performance.

Deciding window size is partly done by reading the table from [6] and comparing with them. They got an accuracy of around 55% with over 500 trails. With some optimization of trying to get even better result, the highest accuracy of the system was recorded at around 60% with 640 trials. Which is very good comparing to the table in the article. However, this does not satisfy the actual aim for the project, to get the hand to close and relax by using the mind. Since a few channels of the data was not working, we could not get the full potential of the data. With all 16 channels working, the result could be even better. The channels not working is an issue with the hardware and is nothing that could fixed during this period of the project. The quality of the data is also affecting the outcome of the project. It depends on a few occasions, for example when sampling the data, the subject had to really focus on relaxing and closing the hand which is easy said than done, and also the environment is also affecting the quality of the data. So, with more time, more data could be provided which could improve the result. Also, since the filter is really bad, we wanted to test the system without the filter completely or improving the current filter, but it was not enough time.

B. Sofia:

When classifying with an SVM in MATLAB the function predict() is used. As of now only one of two outputs is used, the output "label" is used. The other output is "score". In our case, two-class learning, the "score" is a matrix with two columns containing the scores for class one and class two, respectively, for the observations. "Score" in this project could be used for deciding if we want to send a new command to component 3 or not, depending on the score for the prediction. This to attempt to minimize the variation of commands sent. Further, one could also implement an idle state. This is to separate imagining relaxing and not doing anything.

XVII. Discussion - Component 3

(Emanuel and Joakim)

A. Setting up LabVIEW for roboRIO

A lot of time was wasted at this stage. Eventually it worked, but it took too much time. In the end, it seemed obvious to install the software using included DVDs, but since we did not have access to any DVD readers, we really tried to avoid that option until all other options had been tested.

B. Software

The software worked out well. There were no major problems, but smaller problems with, for example, Windows Firewall blocking some packages, but not every time. Most of the issues were created when programming while tired, and not taking a step back and checking if there is an easier solution.

1) UDP Relay - Emanuel: This was not planned for at all, but when it was discovered that roboRIO cannot connect to a LabVIEW socket on the computer, the idea of a relay was quickly made up. Since one group member had previous experience with sockets in Python, a script was written in a few minutes and it worked on the first try.

2) UDP Tester - Emanuel: At first, the system was only tested by changing variables in LabVIEW, although that was enough to test and implement most of the code, it was not enough to test everything. Therefore, a Python script was written to loop through the different commands that can be sent, and sent them to the UDP Relay. It was a good idea that saved a lot of time later on.

3) UDP Receiver - Emanuel: This part was slightly tricky. Mostly because of inexperience with LabVIEW, and it is in general also hard to debug communication since one either receives a message or not, there is no in between. There was also the issue that if this code was run on the roboRIO, it produced a different result than when run on the computer. That was how it was discovered that the roboRIO and the computer are on different networks and that a relay is needed.

4) String to Bits - Emanuel: There were only minor issues with this part because of inexperience with LabVIEW, but it was easily resolved.

5) Main program (and general future work) - Joakim: Because the program allows for not only two states, but a "Between open and closed" state and individual finger combinations, it allows for focus towards further development of the BCI in terms of data acquisition and interpretation. Less focus can be put on the mechanical parts as the hand can take any state that the 3D-printed hand and servos allows for. For further research, it would be better spent time to optimize recordings and recognition of the two states, or for introduction to individual finger movement. Changes to those parts will likely improve the system significantly more than to the robotic hand. That is because they require additional complexity, and that their corresponding research fields likely have more development potential.

However, if one of the main purposes of the robotic hand in the future will be to grip objects of varying sizes and weights, then more research is needed also for the robotic hand. The varying size gripping will require adaptation to additional states, regulating each finger. Perhaps fuzzy logic [59][60] can be implemented to approximate object sizes. Anyhow, these additional states will require much more research in the data acquisition and interpretation fields, because of the limitations that comes with EEG in terms of poor Signal to Noise Ratio (SNR) and neuron reachability. Thus, this may not be a near future work. Gripping objects of different weights is more a question of hardware limitations (servo and wire strength, 3D-print shape and robustness), so this could be a reasonable future work.

6) Angle to PWM - Emanuel: The code was easy to implement but a mistyped number when calculating the linear equations caused one servo to turn, but not fully. Another servo turned in the wrong direction. Once the linear equation was recalculated, one servo started behaving properly, but apparently the sub-VIs of two servos had been linked somehow in a way that caused one servo to turn the wrong direction. Once the issues had been discovered, it was trivial to correct.

C. Hardware

The hardware was mostly easy to work with, but it took more time to design and create than expected. The final results definitely made it worth it.

1) roboRIO: The biggest issues with the roboRIO was that the power supply did not have an obvious way of knowing which lead was positive and which was negative. Great care was taken to make sure that the power supply would not be connected in reverse, and in the end, a multimeter was used to be on the safe side.

2) Servos - Emanuel: There was not much of a choice to be made here. All cheaper servos were out of stock, and the more expensive ones was a lot more expensive. These servos seemed to have all the specifications needed, and they worked better that expected. The pin-out was also the same as the roboRIO.

3) Forearm - Emanuel: The forearm turned out great. There were some issues with SOLIDWORKS trying to create intersecting surfaces that took some time to solve, but the result of the CAD model was very good. There were no issues when printing. The printed forearm cracked because a mounting hole became slightly oval, and while carving with a knife, too much pressure was applied and it cracked. The super glue seems to hold it in place. There should have been an edge in the bottom to give the servos additional support, also the alignment in the top is too good, and the zip ties cannot bend enough to hold them in place. Therefore, the walls around the servos should have been lower, then there would have been no issues. It could be resolved by using thick tape to make the servos thicker, such that the zip ties can get good contact with the servos.

4) Testing component 3 - Emanuel: The accuracy of the delay measurement has a lot of sources of inaccuracy. The biggest being that at 24 frames per second, all that is known is that the arm started moving during the second frame. We do not know if it started moving at the beginning or at the end of the frame. The error is therefore roughly at most $\frac{1}{24} = 0.042$ seconds. Here, the delay was mostly measured to get an

estimate and therefore it does not matter too much. There are other sources of errors, but none should affect the reading as much as the inaccuracy of using low frame rates. Other than that, the test was a good idea since it revealed some errors that otherwise would have been discovered at a later stage, and therefore saved us valuable time.

5) Testing with component 1b and 2b: Everything turned out great, and it was exciting to see the entire system move and work as planned after so much hard work. Component 3 worked as intended.

XVIII. Conclusion Component 1A

A. Fredrik

The combination of LSL and UDP protocols successfully sends packets with a window size of 100 samples. Validation of our system was done by comparing the online data with the recorded offline data shows no-to-minimal package losses.

For future work the data acquisition component must have labels attached to the data sent and include timestamps. Also further development of the user interface is needed, where the subject would have a screen with instructions or images. i.e "grip" or "close hand" and "relax". Preferably an image of a closed hand and then a relaxed open hand. All these are important aspects for a project like this, but were sidelined due to the issues mentioned in the Discussion section. Unity was installed for the user interface, but demanded too much time for a novice that never used it before.

We would highly recommend that the electrodes are tested with a multi-meter and that a connection is established before the headset is put on. The headset is extremely uncomfortable. The pain is subtle and bearable for a while, but when session time is dragged out it becomes unbearable and mentally exhausting and then most likely has a negative effect on the measured data.

XIX. Conclusion Component 2A

A. Victor

The goal of the data analysis was to find the changes in μ frequencies and observe at what time it the changes happening.The wavelet analysis of the data show a suppression in the frequency around 10 Hz as it was expected and the changes is happening after 500 ms of the cue which is due to that the person need some reaction time to start doing the task. The CSP algorithm used in the method showed good results for the training data but less separation in the test data which is a means that the filter is over fitting the data by having a better performance on the test data. This is also shown in the scatter results where the 2 classes overlaps.

B. Inas

This project covers the procedure of implementing a neurorobotic system to control a robotic hand in real-time. For component 2, the task was to receive EEG data from component 1 and process it in real-time using different signal processing techniques into a control command for the robotic hand. The control command had a accuracy of 55%, which means it lacks precision. To improve the accuracy, the train and test data could be split into 80 and 20%, instead of 60 and 40%. Different validation method could also be tested, instead of just one.

C. Simon

The real-time data was sent using a combination of LSL and UDP in order to reduce the amount of lost packets in the connection. The end-to-end time of the packets showed to be sufficiently fast to work in real-time, without creating a stacking delay.

XX. Conclusion Component 1B - Pontus G

The main focus were on producing a system that could sample data both offline and online together with an protocol. The protocol used has a task period of six seconds which gave sufficient data to train on. The offline data were autonomously saved in a recordings folder for OpenBCI that easy could be transferred. The online real-time criteria were tried both with LSL and UDP protocols. LSL exceeding the UDP performance and resulting in the sample time of 0.8s for each package which is what one can expect from a sample frequency of 125Hz.

XXI. Conclusion Component 2B - Sofia

The real-time data interpretation works with the method chosen. After receiving brain activity our system can filter, extract features, and classify the command (closed & relaxed) in around 60% of the cases. The method parameters may not be the optimal ones and require more validation to be considered as good. In pre-processing there are still filter artifacts that can be seen. Also, the window size has been fixed through the project and could be investigated further. CSP sometimes do not separate the data as first expected and needs to be validated more to be considered working. The SVMs kernel is linear but produces a lot of support vectors which could be seen in the results. Further, the kernel choice can still be explored more to fit the output from the CSP better.

XXII. Conclusion - Component 3

A. Joakim

LabVIEW versions other than the 2016 version is not compatible with the RoboRIO interface. In order to use the robotic hand (and thus the whole system), the RoboRIO DVD including LabVIEW version 2016 download is necessary, alternatively new LabVIEW and RoboRIO Toolkit 2016 licenses.

With smaller adjustments to the 3D-printed hand and LabVIEW files, the main file with its associated files allows to receive any instructions and act upon it, in order to grip almost any lightweight object that is possible for a human, or just to move fingers in a one way gripping motion. This allows for disabled patient training, using the robotic hand as visual feedback. Its limitation is to control individual joint movement of the fingers, which can be a whole new project in itself.

B. Emanuel

The hand could control each individual finger based on a command that was received over UDP, allowing for five degrees of freedom. The received command could be read on the computer screen in LabVIEW, along with the current status of every finger. Overall, everything worked as expected. The biggest issue of our hand is the thumb placement. The thumb is placed in such a way that it is difficult to grip some objects. Since the hand design was out of our control, there was nothing we could do about it. Other than that, everything worked out as expected.

1) Future work - Emanuel: 3D print a new hand with better thumb placement and at least one extra degree of freedom to control the location of the thumb sideways. A new hand should also use another sturdier mount to the forearm with at least three mounting points to reduce the risk of cracks if torque is applied. The software and communication protocol could be further developed to allow for any amount of grip on any finger. The mounts for the servos need more support in the bottom, and lower walls in the top.

ACKNOWLEDGMENT

The authors would like to thank Mälardalens University for enabling this project by providing with necessary material and equipment.

REFERENCES

[1] S. Vaid, P. Singh, and C. Kaur, "Eeg signal analysis for bci interface: A review," in *2015 Fifth International Conference on Advanced Computing Communication Technologies*, 2015, pp. 143–147.

[2] C.-F. Lu, S.-J. Tsai, C. Hsiao, H.-M. Lu, C.-W. Jao, P.-S. Wang, and Y.-T. Wu, "Consistency of motor-imagery frequency band is associated with the performance of real-time brain computer interface," in *2019 International Automatic Control Conference (CACS)*, 2019, pp. 1–6.

[3] A. Khorshidtalab and M. J. E. Salami, "Eeg signal classification for real-time brain-computer interface applications: A review," in *2011 4th International Conference on Mechatronics (ICOM)*, 2011, pp. 1–7.

[4] D. Madhuri and P. C. Reddy, "Performance comparison of tcp, udp and sctp in a wired network," in *2016 International Conference on Communication and Electronics Systems (ICCES)*, 2016, pp. 1–6.

[5] "Interpolation techniques," accessed: 2021-12-22. [Online]. Available: http://iridl.ldeo.columbia.edu/dochelp/StatTutorial/Interpolation/index.html

[6] E. Combrisson and K. Jerbi, "Exceeding chance level by chance: The caveat of theoretical chance levels in brain signal classification and statistical assessment of decoding accuracy," *Journal of neuroscience methods*, vol. 250, 01 2015.

[7] "Ml linear discriminant analysis." [Online]. Available: https://www.geeksforgeeks.org/ml-linear-discriminant-analysis/

[8] "6 easy steps to learn naive bayes algorithm with codes in python and r," accessed: 2021-12-22. [Online]. Available: https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/

[9] "Ml - suppoer vector machine(svm)," accessed: 2021-12-22. [Online]. Available: https://www.tutorialspoint.com/machine_learning_with_python/machine_learning_with_python_classification_algorithms_support_vector_machine.htm

[10] "Support vector machines for binary classification," accessed: 2021-12-22. [Online]. Available: https://se.mathworks.com/help/stats/support-vector-machines-for-binary-classification.html

[11] "Validating your machine learning model," accessed: 2021-12-20. [Online]. Available: https://towardsdatascience.com/validating-your-machine-learning-model-25b4c8643fb7

[12] "A gentle introduction to k-fold cross-validation," accessed: 2021-12-20. [Online]. Available: https://machinelearningmastery.com/k-fold-cross-validation/

[13] "Nested cross-validation for machine learning with python," accessed: 2021-12-20. [Online]. Available: https://machinelearningmastery.com/nested-cross-validation-for-machine-learning-with-python/

[14] "Lab streaming layer (lsl)," accessed: 2022-01-08. [Online]. Available: https://docs.openbci.com/Software/CompatibleThirdPartySoftware/LSL/

[15] "What is lsl?" accessed: 2022-01-07. [Online]. Available: https://labstreaminglayer.readthedocs.io/info/intro.html

[16] E. Åstrand, "Time frequency analysis," Lecture in Advanced signal processing, 2021.

[17] M. X. Cohen, *Analyzing neural time series data: theory and practice*. MIT press, 2014.

[18] I. Tomasic, "Bss techniques," Lecture in Advanced signal processing, 2021.

[19] J. Tidare, M. Leon, N. Xiong, and E. Astrand, "Discriminating eeg spectral power related to mental imagery of closing and opening of hand," in *2019 9th International IEEE/EMBS Conference on Neural Engineering (NER)*, 2019, pp. 307–310.

[20] G. Pfurtscheller and C. Neuper, "Motor imagery activates primary sensorimotor area in humans," *Neuroscience Letters*, vol. 239, no. 2, pp. 65–68, 1997. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0304394097008896

[21] J. Long, Y. Li, T. Yu, and Z. Gu, "Target selection with hybrid feature for bci-based 2-d cursor control," *IEEE Transactions on Biomedical Engineering*, vol. 59, no. 1, pp. 132–140, 2012.

[22] J. Tidare, M. Leon, and E. Astrand, "Time-resolved estimation of strength of motor imagery representation by multivariate EEG decoding," *Journal of Neural Engineering*, vol. 18, no. 1, p. 016026, feb 2021. [Online]. Available: https://doi.org/10.1088/1741-2552/abd007

[23] "Ecog performance status - ecog-acrin," ECOG-ACRIN, 2012. [Online]. Available: https://ecog-acrin.org/resources/ecog-performance-status

[24] E.Åstrand, "Lecture 1 in course: Ela411," available at: for students on M taken(2022-01-07).

[25] M. Cook, "Tcp vs. udp: What's the difference?" Lifesize, 10 2017. [Online]. Available: https://www.lifesize.com/en/blog/tcp-vs-udp/

[26] S.-W. Park, J.-H. Bae, J.-H. Park, and M.-H. Baeg, "Development of an anthropomorphic robot hand aimed at practical use for wide service robot application," in *2012 IEEE International Conference on Automation Science and Engineering (CASE)*, 2012, pp. 431–435.

[27] R. Fourie and R. Stopforth, "The mechanical design of a biologically inspired prosthetic hand, the touch hand 3," in *2017 Pattern Recognition Association of South Africa and Robotics and Mechatronics (PRASA-RobMech)*, 2017, pp. 38–43.

[28] B. Gámez, M. Cabrera, L. Serpa, and J. Cabrera, "Mechatronic hand prosthesis for child," in *2015 Asia-Pacific Conference on Computer Aided System Engineering*, 2015, pp. 354–359.

[29] "Parallax Standard Servo (#900-00005)," 2017, accessed: 05-01-2022. [Online]. Available: https://docs.rs-online.com/0e85/0900766b8123f8d7.pdf

[30] "NI roboRIO User Manual (FRC) - National Instruments," 2014, accessed: 05-01-2022. [Online]. Available: https://www.ni.com/pdf/manuals/374474a.pdf

[31] "Hitec HS422 Servo," accessed: 05-01-2022. [Online]. Available: https://media.digikey.com/pdf/data%20sheets/dfrobot%20pdfs/ser0002_web.pdf

[32] "Labview roborio software suite," accessed: 25-11-2021. [Online]. Available: https://www.ni.com/sv-se/support/downloads/software-products/download.labview-roborio-software-suite.html#326406

[33] "Ultracortex "mark iv" eeg headset," accessed: 09-01-2022. [Online]. Available: https://docs.openbci.com/AddOns/Headwear/MarkIV/

[34] "Openbci shop," accessed: 2022-01-08. [Online]. Available: https://shop.openbci.com/collections/frontpage/products/all-in-one-biosensing-r-d-bundle?variant=13043151994952

[35] M. Koctúrová and J. Juhár, "Comparison of dry electrodes for mobile eeg system," accessed: 09-01-2022. [Online]. Available: https://shop.openbci.com/collections/frontpage/products/5-mm-spike-electrode-pack-of-30?variant=8120433606670

[36] A. Morley and L. Hill, "10-20 system eeg placement," accessed: 09-01-2022. [Online]. Available: https://www.ers-education.org/lrmedia/2016/pdf/298830.pdf

[37] "Cyton biosensing board," accessed: 09-01-2022. [Online]. Available: https://docs.openbci.com/Cyton/CytonSpecs/

[38] "Dry eeg comb electrodes," accessed: 09-01-2022. [Online]. Available: https://shop.openbci.com/collections/frontpage/products/5-mm-spike-electrode-pack-of-30?variant=8120433606670

[39] G. Milsap and P. Peranich, "Technical note: A low-cost research platform for brain-computer-interface applications in mixed reality," in *2021 10th International IEEE/EMBS Conference on Neural Engineering (NER)*, 2021, pp. 515–518.
[40] "Pic32mx250f128b datasheet (pdf) - microchip technology," accessed: 09-01-2022. [Online]. Available: https://pdf1.alldatasheet.com/datasheet-pdf/view/516001/MICROCHIP/PIC32MX250F128B.html
[41] "The openbci gui," accessed: 09-01-2022. [Online]. Available: https://docs.openbci.com/Software/OpenBCISoftware/GUIDocs/
[42] B. Blankertz, G. Dornhege, M. Krauledat, K.-R. Müller, and G. Curio, "The non-invasive berlin brain–computer interface: fast acquisition of effective performance in untrained subjects," *NeuroImage*, vol. 37, no. 2, pp. 539–550, 2007.
[43] J. Long, Y. Li, H. Wang, T. Yu, J. Pan, and F. Li, "A hybrid brain computer interface to control the direction and speed of a simulated or real wheelchair," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 20, no. 5, pp. 720–729, 2012.
[44] S. Lemm, B. Blankertz, G. Curio, and K.-R. Muller, "Spatio-spectral filters for improving the classification of single trial eeg," *IEEE transactions on biomedical engineering*, vol. 52, no. 9, pp. 1541–1548, 2005.
[45] "The openbci gui," accessed: 2022-01-09. [Online]. Available: https://docs.openbci.com/Software/OpenBCISoftware/GUIDocs/
[46] "Matrices and arrays," accessed: 2022-01-08. [Online]. Available: https://se.mathworks.com/help/matlab/matrices-and-arrays.html
[47] "Welcome to the openbci community," accessed: 2022-01-07. [Online]. Available: https://docs.openbci.com/
[48] "Openbci_gui networking output guide," accessed: 2022-01-07. [Online]. Available: https://docs.google.com/document/d/e/2PACX-1vR_4DXPTh1nuiOwWKwIZN3NkGP3kRwpP4Hu6fQmy3jRAOaydOuEI1jket6V4V6PG4yIG15H1N7oFfdV/pub
[49] H. Ramoser, J. Muller-Gerking, and G. Pfurtscheller, "Optimal spatial filtering of single trial eeg during imagined hand movement," *IEEE Transactions on Rehabilitation Engineering*, vol. 8, no. 4, pp. 441–446, 2000.
[50] Y. Wang, S. Gao, and X. Gao, "Common spatial pattern method for channel selelction in motor imagery based brain-computer interface," in *2005 IEEE Engineering in Medicine and Biology 27th Annual Conference*, 2005, pp. 5392–5395.
[51] "cov," accessed: 06-01-2022. [Online]. Available: https://se.mathworks.com/help/matlab/ref/cov.html
[52] "svd," accessed: 06-01-2022. [Online]. Available: https://se.mathworks.com/help/matlab/ref/double.svd.html
[53] "eig," accessed: 06-01-2022. [Online]. Available: https://se.mathworks.com/help/matlab/ref/eig.html
[54] J. Cervantes, F. Garcia-Lamont, L. Rodriguez-Mazahua, and A. Lopez, "A comprehensive survey on support vector machine classification: Applications, challenges and trends," *Neurocomputing*, vol. 408, pp. 189–215, 2020.
[55] "fitcsvm," accessed: 07-01-2022. [Online]. Available: https://se.mathworks.com/help/stats/fitcsvm.html
[56] "Hs-422 deluxe standard servo," accessed: 2022-01-06. [Online]. Available: https://hitecrcd.com/products/servos/sport-servos/analog-sport-servos/hs-422/product
[57] "Activity 4: Test speed control," accessed: 08-12-2021. [Online]. Available: https://learn.parallax.com/tutorials/robot/shield-bot/robotics-board-education-shield-arduino/chapter-3-assemble-and-test-5
[58] "Hitec hs-422," accessed: 2022-01-06. [Online]. Available: https://grabcad.com/library/hitec-hs-422-1
[59] "Artificial Intelligence - Fuzzy Logic Systems," accessed: 09-01-2022. [Online]. Available: https://www.tutorialspoint.com/artificial_intelligence/artificial_intelligence_fuzzy_logic_systems.htm
[60] L. A. Zadeh, "Fuzzy logic," 1988, accessed: 09-01-2022. [Online]. Available: https://ieeexplore.ieee.org/document/53