JST: Smart Systems and Devices
Volume 34, Issue 3, September 2024, 052-059

Low-Cost Effective Hands-Free Control Devices for Quadriplegia

Son T. Nguyen**, Manh X. Vuong, Tien D. Nguyen
Hanoi University of Science and Technology, Ha Noi, Vietnam
**Corresponding author email: son.nguyenthanh@hust.edu.vn*

Abstract

Hands-free control devices are very beneficial for people with quadriplegia. By using these devices, severely disabled people can obtain much more independence and significantly reduce the assistance from relatives. Hands-free devices can be developed based on head movement detection, eye blinking detection, and speech recognition. For many years, these hands-free devices have been very costly and even ineffective for many users. In addition, they cannot adapt to different kinds of users. In this study, low-cost, efficient, hands-free devices have been proposed with the use of inexpensive hardware and software. Firstly, a head-direction-based control system is formed by using an ADXL335 accelerometer and an Arduino Uno board. In this system, intentional head movement can be detected by using a feedforward neural network. An eye-blink-based system can be developed by using a MindWave mobile headset and an Arduino Uno board. An effective speech recognition system has been developed using an Arduino Nano 33 BLE Sense board with a speech recognition technique that does not require any learning model. Finally, these hands-free control devices are not only effective but also very affordable for various kinds of severely disabled users.

Keywords: Quadriplegia, hands-free control devices, head movement-based control, eyeblink-based control, speech recognition-based control.

1. Introduction

Quadriplegia, sometimes known as tetraplegia, is a type of severe disability affecting all the limbs and body from the neck downward. The main cause of quadriplegia is a severe spinal cord injury. Other conditions, including stroke, cerebral palsy, traffic, and work accidents, are also possible causes of quadriplegia. As people with quadriplegia have been unable to use conventional input devices such as a hand-operated joystick, a computer keyboard or a mouse, there are various developments of alternative hands-free access systems based on voice [1, 2], face direction [3], head movement [4], brainwave (EEG) [5, 6].

For a long period, voice recognition has been a topic of much research. Using speech recognition techniques, disabled people can use their voices to control the moving direction of power wheelchairs [1]. Voice recognition systems can be categorized into two types: Speaker-dependent and speaker-independent systems Speaker-independent systems can respond to a wide range of speech patterns since they respond to a word regardless of who is speaking. For applications where many individuals utilize the same system, this kind of voice recognition technology is required. Although voice recognition has been commercially available and has been also integrated into many current electronic devices, voice-based control systems are usually ineffective when they operate in cluttered environments. In such areas, the quality of voice recognition is significantly affected by noise and channel distortions. The noise phenomenon not only affects the acoustics of the speed signal but also distorts it at the source. This problem is extremely difficult to overcome, especially when there is no prior knowledge of the noise or distortions.

For disabled people with severe spinal cord injuries, their head movement can be used to access assistive devices, including power wheelchairs [4]. The head movement-based control systems can be formed by using a tilt sensor mounted on a cap worn by the user and a data acquisition device to record the head movement signal, which will be later processed by a computer to output the desired commands to activate home appliances. The main advantage of these systems is that they can be conveniently developed with various control algorithms.

Many studies have focused on the Brain-Computer Interface (BCI) to obtain communication between humans and machines using brain signals [5, 6]. A BCI is formed with a Graphic User Interface (GUI) convenient for monitoring and analyzing the brain signal. A BCI can also be used to generate desired commands to control peripheral devices. However, all BCI projects have been developed with very costly hardware and software. Due to the complex

ISSN 2734-9373
https://doi.org/10.51316/jst.176.ssad.2024.34.3.7
Received: Apr 2, 2024; revised: May 5, 2024,
accepted: May 5, 2024

and time-varying properties of brain signals, BCI research is a multidisciplinary field integrating researchers from neuroscience, psychology, engineering, computer science, and rehabilitation. Since the intentions of a person can be directly conveyed through a BCI, this technology is expected to bring a lot of benefits for severely disabled people in the next few years.

This research proposes three hands-free control solutions based on head movement, eye blink and speech. They can be built very conveniently with inexpensive hardware and software. These hands-free assistive devices can be affordable for many low-income disabled people. The structure of the paper is organized as follows: Section 2 presents the feedforward neural network for multi-class classification tasks. In Section 3, an intelligent head-movement based control system is described in detail. An eye-blink based hands-free control system is mentioned in Section 4. A speech recognition based hands-free control system is illustrated in Section 5. Finally, the conclusion of this research is stated in Section 6.

2. Feedforward Neural Network for Classification

The feedforward neural network (FFNN) is probably the most widely used in numerous engineering applications. In a FFNN, a series of layers is fully connected from the inputs to the outputs without any feedback. The network training can be performed by using the backpropagation algorithm. In this study, the application of the FFNN has been restricted to a FFNN with a single hidden layer for multi-class classification tasks.

2.1. Forward Propagation

The input values of the network are denoted x_i where $i = 1,...,d$. The inputs, x_i , are used to compute the activations of the first hidden layer as follows:

$$a_j^{(1)} = w_{ji}^{(1)} x_i + b_j^{(1)} \tag{1}$$

where $w_{ji}^{(1)}$ are the weight on the connection from the i -th input unit in the input layer to the j -th unit in the hidden layer. $b_j^{(1)}$ are the bias of the j -th unit in the hidden layer. Then the activation $a_j^{(1)}$ are used to compute the outputs of the first hidden layer as follows:

$$y_j^{(1)} = f_1\left(a_j^{(1)}\right) = \tanh\left(a_j^{(1)}\right) \tag{2}$$

Here $f_1(.)$ is called the activation function of the units in the hidden layer. The partial derivative of $y_j^{(1)}$ with respect to $a_j^{(1)}$ has the following form:

$$\frac{\partial y_j^{(1)}}{\partial a_j^{(1)}} = 1 - \left(\tanh\left(a_j^{(1)}\right)\right)^2 = 1 - \left(y_j^{(1)}\right)^2 \tag{3}$$

The outputs of the hidden layer, $y_j^{(1)}$, are used to calculate the activations of the nodes in the second hidden layer as follows:

$$a_k^{(2)} = w_{kj}^{(2)} y_j^{(1)} + b_k^{(2)} \tag{4}$$

where $w_{kj}^{(2)}$ are the weight on the connection from the j -th unit in the hidden layer to the k -th unit in the output layer. $b_k^{(2)}$ are the bias of the j -th unit in the output layer. Then the activation $a_k^{(2)}$ are used to compute the outputs of the output layer by using a "softmax" activation function as follows:

$$z_k = \frac{\exp(a_k^{(2)})}{\sum_{k'=1}^{c} \exp(a_{k'}^{(2)})} \tag{5}$$

2.2. Data Error Function

For c -class ($c > 2$) classification problems, the neural network training is a process to minimize an error function having the following form:

$$E_D = -\sum_{n=1}^{N} \sum_{k=1}^{c} t_k^n \ln z_k^n \tag{6}$$

in which z_k^n is the k -th output corresponding to the n -th training pattern and t_k^n is the k -th target corresponding to the n -th training pattern. E_D is also known as the "entropy" function.

2.3. Backpropagation

The backpropagation is the most common technique to evaluate partial derivatives of objective functions defined in the neural network training. The first step in evaluating the derivatives of the error function is to perform a forward propagation for the entire data set to obtain the activation of the nodes in the output layer. Then the partial derivative of the error function E_D with respect to $a_k^{(2)n}$ is given by:

$$\frac{\partial E_D^n}{\partial a_k^{(2)n}} = z_k^n - t_k^n \tag{7}$$

To evaluate the derivatives of E_D with respect to the weights and biases on the connection from the second hidden layer to the output layer, the error of the output units on the n -th pattern is firstly calculated as follows:

$$\delta_k^{(2)n} = z_k^n - t_k^n \tag{8}$$

The derivatives of E_D with respect to the weights and biases on the connection from the hidden layer to the output layer are given by:

$$\frac{\partial E_D}{\partial w_{kj}^{(2)}} = \sum_{n=1}^{N} \delta_k^{(2)n} y_j^{(2)n} \tag{9}$$

The derivatives of E_D with respect to the biases of the units in the output layer are given by:

$$\frac{\partial E_D}{\partial b_k^{(2)}} = \sum_{n=1}^{N} \delta_k^{(2)n} \tag{10}$$

To find the derivatives of E_D with respect to the weights and biases on the connection from the input layer to the hidden layer, the errors $\delta_k^{(2)n}$ are firstly back propagated through the second layer to obtain error signals for the units in the second hidden layer as follows:

$$\begin{aligned}\delta_j^{(1)n} &= g'\left(a_j^{(1)n}\right)\sum_{k=1}^{c} w_{kj}^{(2)}\delta_k^{(2)n} \\ &= \left(1-\left(y_k^{(2)n}\right)^2\right)\sum_{k=1}^{c} w_{kj}^{(2)}\delta_k^{(2)n}\end{aligned} \tag{11}$$

The derivatives of E_D with respect to the weights on the connection from the input layer to the hidden layer are given by:

$$\frac{\partial E_D}{\partial w_{ji}^{(1)}} = \sum_{n=1}^{N} \delta_j^{(1)n} x_j^n \tag{12}$$

The derivatives of E_D with respect to the biases of the hidden layer are given by:

$$\frac{\partial E_D}{\partial b_j^{(1)}} = \sum_{n=1}^{N} \delta_j^{(1)n} \tag{13}$$

2.4. Network Regularization

To prevent the network after being trained from overfitting, a network regularization is usually required. A weight decay is added to the error function to obtain a cost function as follows:

$$S = E_D + E_W \tag{14}$$

in which E_W is called the "weight decay" to penalize massive values of weights and biases probably causing the overfitting phenomenon.

$$E_W = \frac{\alpha}{2}\|w\|^2 \tag{15}$$

where α is known as the regularization factor or is sometimes called the "hyperparameter". w is the weights and biases. The value of α should be determined appropriately for each classification problem. The weights and biases are finally updated at each iteration as follows:

$$w_{ji}^{(1)} \leftarrow w_{ji}^{(1)} - \eta\left(\frac{\partial E_D}{\partial w_{ji}^{(1)}} + \alpha w_{ji}^{(1)}\right) \tag{16}$$

$$\partial b_j^{(1)} \leftarrow \partial b_j^{(1)} - \eta\left(\frac{\partial E_D}{\partial b_j^{(1)}} + \alpha b_j^{(1)}\right) \tag{17}$$

$$w_{kj}^{(2)} \leftarrow w_{kj}^{(2)} - \eta\left(\frac{\partial E_D}{\partial w_{kj}^{(2)}} + \alpha w_{kj}^{(2)}\right) \tag{18}$$

$$b_k^{(2)} \leftarrow b_k^{(2)} - \eta\left(\frac{\partial E_D}{\partial b_k^{(2)}} + \alpha b_k^{(2)}\right) \tag{19}$$

3. Head-Movement Based Hands-Free Control

A hands-free control system based on head movement and FFNN consists of the following parts:

- An ADXL335 accelerometer is mounted on a cap of a person to sense his/her head tilt according to the *x* and *y* axes.
- An Arduino Uno R3 board is used to convert specific intentional head movements at analogue inputs A0 and A1 into appropriate control commands at digital outputs 7, 8, 12 and 13. The mapping from the analogue inputs to the digital outputs of the Arduino Uno board is performed by a pre-trained FFNN.
- A relay module is used to make an interface between the digital outputs of the Arduino Uno board and actuators such as home appliances or assistive devices.
- A single output switching power supply for energizing the relay module.

To deploy a control algorithm using a FFNN, head movement data were firstly collected from the author of this paper. For a computer-based system, the Arduino IO Package from the MathWorks is used for handling the interface between Simulink and the Arduino Uno board [7]. For a standalone system, the Simulink Support Package for Arduino Hardware [8] can be used.

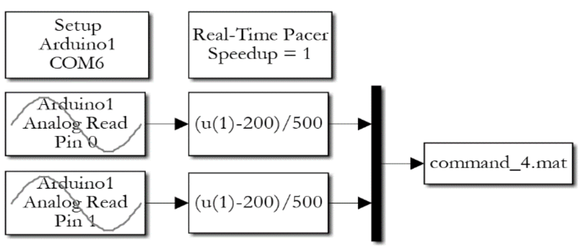

Fig. 1. Simulink diagram for acquiring head movement data.

Fig. 1 shows a Simulink diagram for acquiring head movement signals from analogue inputs A0 and A1 of the Arduino Uno board. The measured head movement signals are then normalized to obtain the values ranging from 0 to 1 by using the following equations:

$$x = \frac{ADC_x - 200}{500} \tag{20}$$

$$y = \frac{ADC_y - 200}{500} \tag{21}$$

in which ADC_x and ADC_y are readings from analogue inputs A0 and A1 of the Arduino Uno board, respectively. As analogue to digital converters (ADCs) of the Arduino Uno board is 10 bits, the readings from the analogue pins vary from 0 to 1023 corresponding to the input voltage of the ADCs from 0 to 5V.

In this study, only four head-movement-based commands are considered corresponding to forward, backward, left, and right head movements. The status of not moving the head can be seen as the neutral state. The duration of for collecting each head movement pattern was 3 seconds with a sampling period of 100 milliseconds.

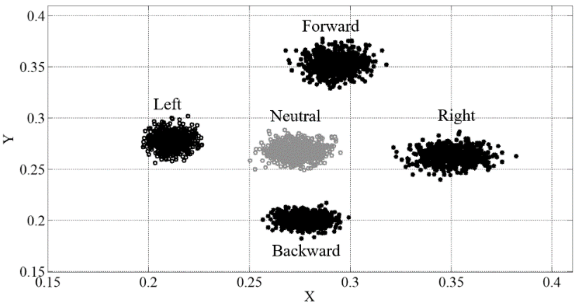

Fig. 2. Head movement patterns in a 2D plane.

A large amount of head movement data can be formed by adding some noise to acquired head movement data. Then, the final data can be seen as a dataset collected from many people, and it has some degree of uncertainty. Normalized head movement patterns can be visualized in a 2D plane as shown in Fig. 2. There are five types of head movement patterns corresponding to the neutral position of the head, forward, backward, left and right head tilts.

To differentiate between the natural and voluntary head movements, the head movement signals are converted into a vector of 20 elements corresponding to 10 elements for x axis and 10 elements for y axis, respectively. Fig. 3 displays the conversion of the head movement signals into a vector of 20 elements for the neural network inputs using the unit delays. To activate or control assistive devices, digital outputs 7, 8, 12 and 13 of the Arduino Uno board are used as depicted in Fig. 4.

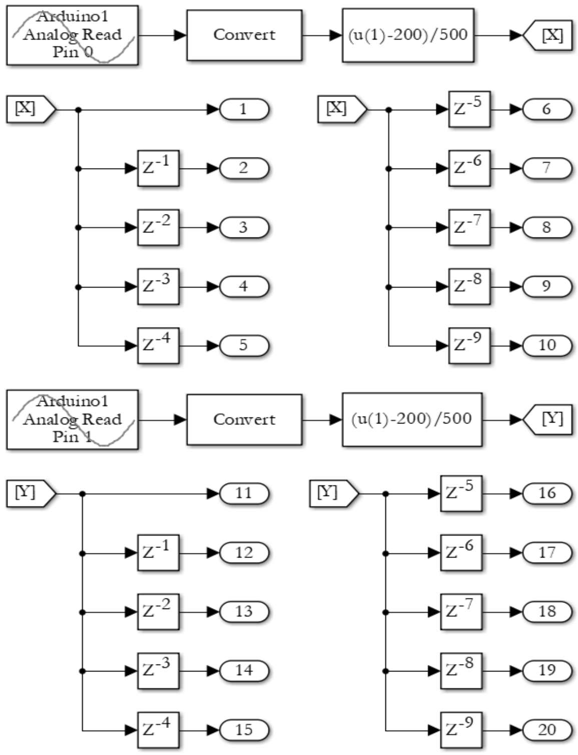

Fig. 3. Simulink diagram of a 20-element vector of the normalized head movement signal in the computer-based system.

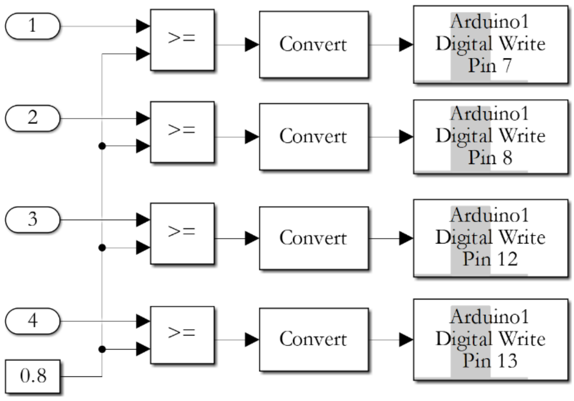

Fig. 4. Simulink diagram for controlling digital outputs of the Arduino Uno board in the computer-based system.

The FFNN used to classify head movement has the following architecture:

- 20 inputs, corresponding to 10 elements from x axis and 10 elements from y axis and one augmented input assigned to be one,
- 4 hidden units,
- 5 outputs, corresponding to stop, forward, backwards, left, and right classes.

The training procedure was then carried out as follows:

- The weights and biases of the network were randomly initialized.
- A suitable cost function was established and minimized using the gradient descent optimization technique. The learning rate and the regularization parameter were chosen to be 0.001 and 0.1, respectively. The maximum number of iterations is set to 10000.
- As the number of network training iterations reaches a specified value, the network training procedure is terminated.

The MATLAB code for training the FFNN has the following form:

```
load head_movement_data.mat;
N = length(xa);
nin = 20;
nhidden = 4;
nout = 5;
w1 = randn(nin, nhidden);
b1 = randn(1, nhidden);
w2 = randn(nhidden, nout);
b2 = randn(1, nout);
M = 10000;
nu = 0.001;
alpha = 0.1;
for i = 1:M
   a1 = xa*w1 + ones(N,1)*b1;
   y = tanh(a1);
   a2 = y*w2 + ones(N,1)*b2;
   temp = exp(a2);
   z = temp./(sum(temp, 2)*ones(1,nout));
   delout = z - ta;
   gw2 = y'*delout;
   gb2 = sum(delout,1);
   delhid  = (delout*w2').*(1.0 - y.*y);
   gw1 = xa'*delhid;
   gb1 = sum(delhid,1);
   w1 = w1 - nu*(gw1 + alpha*w1);
   b1 = b1 - nu*(gb1 + alpha*b1);
   w2 = w2 - nu*(gw2 + alpha*w2);
   b2 = b2 - nu*(gb2 + alpha*b2);
end
```

The MATLAB embedded function of the pre-trained FFNN has the following form:

```
function [z2, z3, z4, z5] = fcn(x1, x2, x3,x4,...
x5, x6, x7, x8, x9, x10, x11, x12,...
x13, x14, x15, x16, x17, x18, x19, x20)
x = [x1 x2 x3 x4 x5 x6 x7 x8 x9 x10 x11 x12...
    x13 x14 x15 x16 x17 x18 x19 x20];
w1 = [
  -1.0701   -1.1699   -1.7683   -0.6766
  -1.1969   -1.4613   -0.9168   -0.3645
  -0.9041   -1.4490   -1.6401   -0.2300
  -1.2951   -1.4908   -1.5959   -0.0227
  -0.8506   -0.9116   -1.8159   -0.7094
  -1.2478   -0.6224   -1.1037   -0.0399
  -1.1355   -0.6380   -1.9169    0.4631
  -0.8509   -1.7671   -1.0207   -0.0956
  -1.1010   -1.3963   -1.4225    0.0984
  -1.9776   -1.4725   -1.6050   -0.3917
  -0.8067   -0.4495    2.3159    0.0406
  -0.4535   -1.0733    1.5734    0.1064
  -1.0724   -0.5000    1.9324   -0.1223
  -0.0022   -0.7477    1.6417   -0.3098
  -0.2996   -0.6414    1.7366    0.3786
  -0.8285   -1.3450    1.5677   -0.5338
  -0.8074   -0.7650    2.3062   -0.7214
  -0.0002   -0.9299    1.6393    0.3422
  -0.9063   -0.8138    2.0875   -0.1077
  -0.7788   -1.1662    1.6798   -2.4875
   ];
b1 = [4.4796    5.8904   -0.8160   -0.6908];
w2 = [
  -1.5749   -2.5011    2.8272    4.5946   -2.5678
   1.3914   -4.8625    3.5258    4.1807   -4.8723
   0.4345    5.4099   -7.1124    5.7713   -4.6906
  -1.4481    0.5707    0.3449    0.4430    0.5213
   ];
b2 = [1.8720   -0.2482    0.1748   -0.3715    0.2756];
a1 = x*w1 + b1;
y1 = tanh(a1);
a2 = y1*w2 + b2;
z = exp(a2)/sum(exp(a2));
z2 = z(2);
z3 = z(3);
z4 = z(4);
z5 = z(5);
```

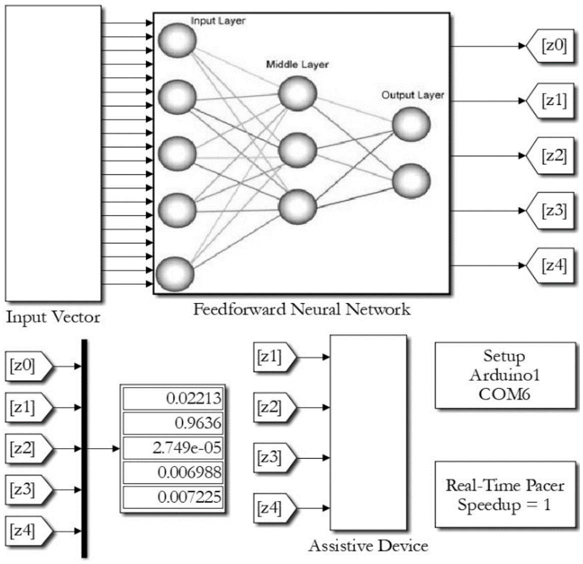

Fig. 5. GUI of the computer-based system.

Fig. 5 shows a Simulink diagram for the entire computer-based control system in which a graphic user interface (GUI) can allow the user's system to monitor the control output signals in real-time.

4. Eye-Blink Based Hands-Free Control

Since giving some head movements could be difficult for many people with quadriplegia, an eye-blink-based control system is developed in this study.

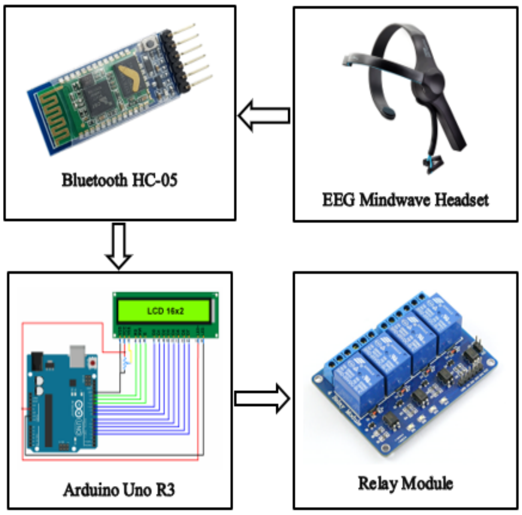

Fig. 6. Hardware of the eye blinking-based control unit.

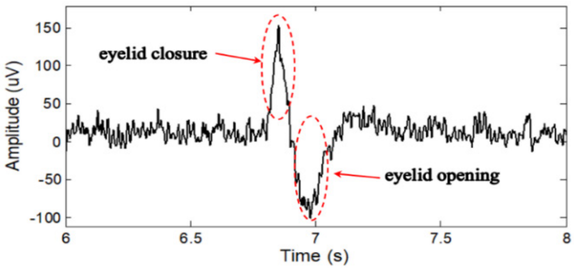

Fig. 7. EEG signal with eyelid opening and eyelid closure.

Fig. 6 shows an eye-blink-based control system. The system is formed using a MindWave Mobile EEG Headset, which is a low-cost single-electrode EEG device produce by NeuroSky [9]. Using this EEG device, the brain signal can be wirelessly transmitted to an Arduino Uno R3 board using a Bluetooth HC-05 module.

Fig. 7 is a raw EEG signal of a single-eye blink corresponding to an eyelid opening and an eyelid closure. At the time of the eyelid closure and the eyelid opening, there are significant changes in the amplitude of the raw EEG signal. A single eye blink can be detected if there is no next eye blink in a long time. Whereas a double blink can be only detected with two single-eye blinks in a short time.

The Arduino Uno R3 board was programmed to extract intentional double-eye blinks used to change commands to access assistive devices or activate electrical equipment via a relay module. A flowchart of the algorithm for choosing one of the two commands based on the detection of double-eye blinks is shown in Fig. 8.

The status of the eye blink can be monitored by either a computer-based system or a standalone system. For the computer-based system, information of the signal quality and the number of eye blinks can be displayed through the Serial Monitor of the Arduino IDE. For a standalone system, a liquid crystal display (LCD) should be used to help the system's user monitor the blinks detected and the current commands. As the user knows the current command, they will be able to give an intentional double blink to select another command.

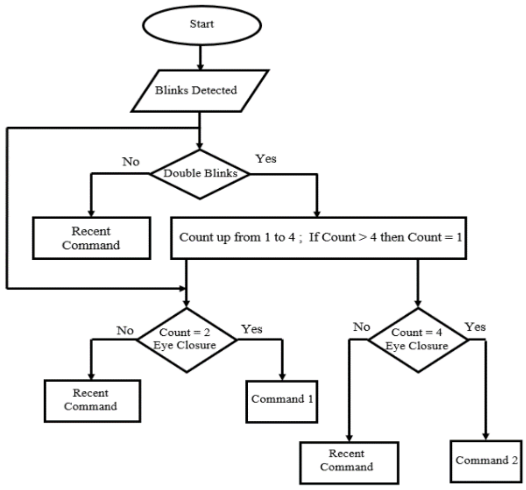

Fig. 8. Flowchart of the algorithm to select two commands using eye blinking.

The C code for changing two commands based on intentional eye blinks extracted from the raw EEG has the following form:

```
#include <Mindwave.h>
Mindwave mindwave;
int Quality = 0;
int Blink = 0;
int Count = 0;
int Mode = 0;
int Command = 0;

void setup() {
     Serial.begin(57600);
     pinMode(7, OUTPUT);
     pinMode(8, OUTPUT);
}

void Mode_1() {
     digitalWrite(7, HIGH);
     digitalWrite(8, LOW);
}

void Mode_2() {
      digitalWrite(7, LOW);
      digitalWrite(8, HIGH);
}

void onMindwaveData(){
     Quality = mindwave.quality();
     Blink = mindwave.blink();
     Count = Count + 1;
     if (Count > 4)
          Count = 0;
     if ((Count == 2)&&(Blink == 2))
          Command = 1;
```

```
        if ((Count == 4)&&(Blink == 2))
            Command = 2;
        Serial.print("Quality:");
        Serial.print(mindwave.quality());
        Serial.print("\tBlink:");
        Serial.print(mindwave.blink());
        Serial.print("\tCount:");
        Serial.print(Count);
        Serial.print("\tCommand:");
        Serial.println(Command);
}

void loop() {
        mindwave.update(Serial,onMindwaveData);
}
```

5. Voice Based Hands-Free Control

Today, many voice-based control devices have been commercially available. In embedded speech solutions, Cyberon Corporation (Taiwan) offers a lot of speed recognition and text-to-speech technologies. Speech solutions developed by Cyberon Corporation can be widely used for mobile and portable devices to provide customers with a convenient, natural, and reliable user experience [10]. In this section, a development of an effective speech recognition system using an Arduino Nano 33 BLE Sense, and DSpotter which is a speech recognition product of Cyberon Corporation is described in detail.

5.1. The Arduino Nano 33 BLE Sense Board

The Nano 33 BLE Sense board, as shown in Fig. 9 is Arduino's 3.3V AI-enabled board. It comes with a series of embedded sensors including:

- Nine-axis inertial sensor ideal for wearable devices,
- Humidity, and temperature sensor for getting highly accurate measurements of the environmental conditions,
- Barometric sensor for making a simple weather station,
- Microphone for capturing and analyzing sound in real-time,
- Gesture, proximity, light color, and light intensity sensors for estimating the room's luminosity, but also whether someone is moving close to it.

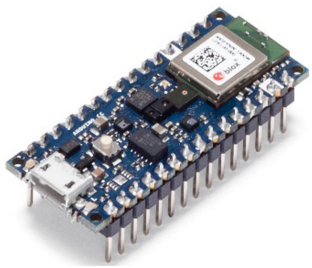

Fig. 9. Arduino Nano 33 BLE Sense board.

5.2. The Arduino Speech Recognition Engine

The Arduino Speech Recognition Engine, developed by Cyberon Corporation, is one of the best available platforms to perform speech recognition on multiple embedded microcontrollers, including the Arduino Nano 33 BLE Sense board. To get a demo license, users must access the following website: https://tool.cyberon.com.tw/ArduinoDSpotterAuth/CTMain.php

The users need to perform the following steps:

- Configurating a model (Fig. 10),
- Creating/Importing a customized project (Fig. 11),
- Creating a trigger word (Fig. 12),
- Creating command words (Fig. 13),
- Reviewing and confirming the customized project (Fig. 14).
- Checking the email sent by Cyberon and downloading a set of files like:
- CybLicense_1712044307.h
- Model_1712044307.h
- Model_1712044307_Arduino_33ble.dsproj
- Info_1712044307.txt

These files must be in the same directory with the Arduino project file. In the Arduino file, update the new license and model as follows:

#include "CybLicense_1712044307.h"

#include "Model_1712044307.h"

From now, speech recognition tasks can be performed.

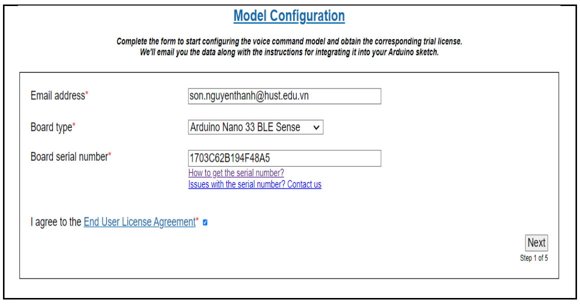

Fig. 10. Model configuration.

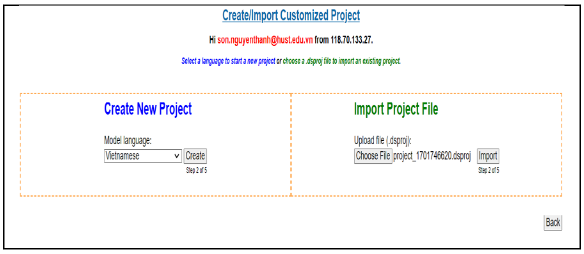

Fig. 11. Creating and importing customized project.

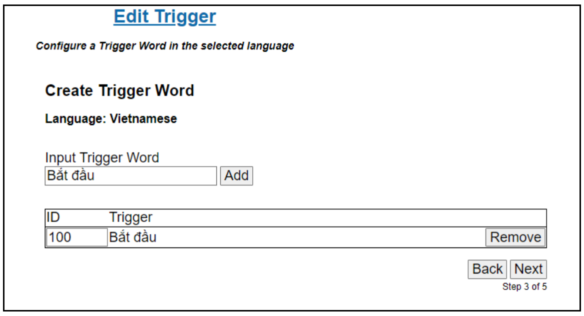

Fig. 12. Creating a trigger word.

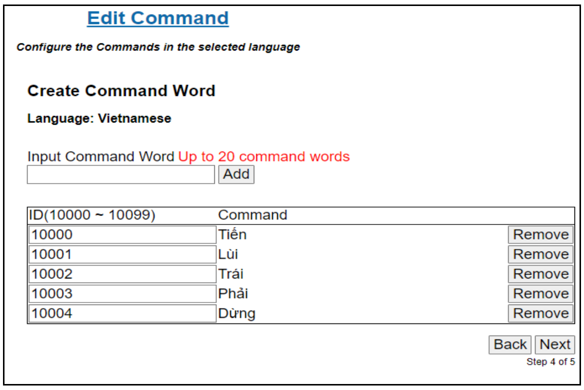

Fig. 13. Creating command words

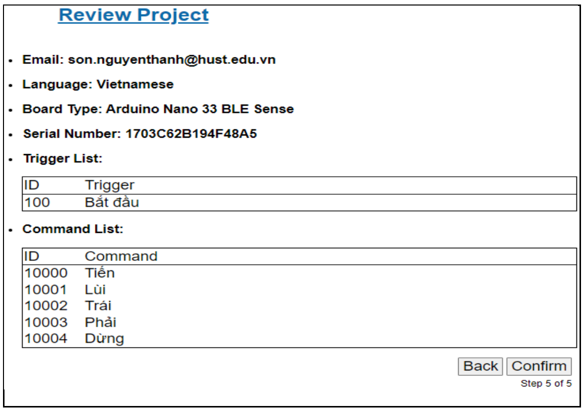

Fig. 14. Reviewing project.

6. Conclusion

The study has demonstrated three effective hands-free solutions using head movement, eye blink extracted from the brain wave signal, and a speech recognition system independently with its users. As these control methods are low-cost, they can be conveniently deployed to be affordable with various kinds of quadriplegic people.

Acknowledgments

This research is funded by Hanoi University of Science and Technology (HUST) under project number T2023-PC-041.

References

[1] S. Poirier, F. Routhier, A. Campeau-Lecours, Voice control interface prototype for assistive robots for people living with upper limb disabilities, IEEE 16th International Conference on Rehabilitation Robotics (ICORR), Toronto, Canada, Jun. 24-28, 2019, pp. 46-52. https://doi.org/10.1109/ICORR.2019.8779524

[2] S. Umchid, P. Limhaprasert, S. Chumsoongnern, T. Petthong, T. Leeudomwong, Voice controlled automatic wheelchair, 11th Biomedical Engineering International Conference (BMEiCON), Chiang Mai, Thailand, Nov. 2018. https://doi.org/10.1109/BMEiCON.2018.8609955

[3] N. Kawarazaki, M. Yamaoka, Face tracking control system for wheelchair with depth image sensor, 13th International Conference on Control Automation Robotics & Vision (ICARCV), Singapore, Dec. 10-12, 2014, pp. 781-786. https://doi.org/10.1109/ICARCV.2014.7064403

[4] Son T. Nguyen, Hung T. Nguyen, Philip B. Taylor, James Middleton, Improved head direction command classification using an optimised Bayesian neural network, 2006 International Conference of the IEEE Engineering in Medicine and Biology Society, New York, USA, 30 Aug. 2006-03 Sep. 2006, pp. 5679-5682. https://doi.org/10.1109/IEMBS.2006.260430

[5] S. K. Swee, L. Z. You, Fast Fourier analysis and EEG classification brainwave-controlled wheelchair, 2016 2nd International Conference on Control Science and Systems Engineering (ICCSSE), Singapore, Jul. 27-29, 2016, pp. 20-23. https://doi.org/10.1109/CCSSE.2016.7784344

[6] W. Zgallai, J. T. Brown, A. Ibrahim, F. Mahmood, K. Mohammad, M. Khalfan, Deep learning AI application to an EEG driven BCI smart wheelchair, 2019 Advances in Science and Engineering Technology International Conferences (ASET), Dubai, United Arab Emirates, 26 Mar-10 Apr. 2019. https://doi.org/10.1109/ICASET.2019.8714373

[7] The ArduinoIO Package from the MathWorks: [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/32374-legacy-matlab-and-simulink-support-for-arduino?s_cid=srchtitle

[8] Simulink Support Package for Arduino Hardware, [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/40312-simulink-support-package-for-arduino-hardware

[9] NeuroSky, [Online]. Available: https://store.neurosky.com

[10] Cyberon Corporation, [Online]. Available: https://www.cyberon.com.tw/index.php?lang=en