Advances In Natural And Applied Sciences

2020 May-August 14(2): pages 287-297
DOI: 10.22587/anas.2020.14.2.40

Research Article AENSI Publications

Quadcopter Control using LabVIEW for a Brain Computer Interface System

Haider Alwasiti[1], **Mohd Zuki Yusoff**[2]

[1]Department of Electrical and Electronic Engineering, Universiti Teknologi Petronas, 32610 Bandar Seri Iskandar, Perak, Malaysia
[2]Department of Electrical and Electronic Engineering, Universiti Teknologi PETRONAS, 32610 Bandar Seri Iskandar, Perak, Malaysia

Correspondence Author: Haider Alwasiti, Department of Electrical and Electronic Engineering, Universiti Teknologi Petronas, 32610 Bandar Seri Iskandar, Perak, Malaysia
E-mail: *wasiti14@gmail.com*

Received date: 22 May 2020 , **Accepted date:** 22 July 2020, **Online date:** 22 August 2020

Abstract

A method to control a flying quadcopter robot using LabView G-language is being demonstrated in this paper. The controller is designed to be interfaced with the BCI algorithm that is providing a system for controlling a quadcopter using the thinking process of the user. The whole system is aiming to develop a novel method to increase the accuracy and robustness of the BCI control. The modular approach that has been developed using Labview has shown the advantage of simplicity and flexibility to control quadcopters, which can be used to be interfaced with brain computer interface systems, which allows users to control a quadcopter by the brain's thinking process.

Keywords: EEG; Quadcopter; BCI

INTRODUCTION

For the past two decades, brain computer interface (BCI) became a very active research topic all over the world [1]. It is aimed to provide the brain with another route of communication and control other than the normal pathway of controlling the muscles of the body. By intentionally modifying (modulating) the electrical signals of the brain using an exogenous (e.g., visual or auditory stimuli) or endogenous (e.g., thinking to move part of the body) source of modifier of the electrical signals of the user's brain, it is possible to translate those changes into a specific control task outside of the body [1-3]. The present work aims to add a controller for current BCI systems. The BCI system is translating the intentions of the user into higher abstract commands, and the controller is taking care of the details of controlling the quadcopter to carry out the intended tasks for the user.

There are certain medical conditions that can affect the CNS in a specific pathological condition resulting in losing the brain's ability to control the muscles. One of the well-known diseases called locked-in syndrome [4]. Some of them develop a complete loss of control of all the muscles of the body, a condition called total locked-in syndrome. For such patients there is no way to make any communication nor controlling any of the muscles of the body. They are locked-in inside their mind. Any method which helps to use their thinking process to control outside devices directly without using the normal physiological route to control or communicate with others is alleviating a tremendous distress.

The research in the field of BCI did not yield any robust system yet to be used widely clinically and commercially. The current methods of BCI research aim to modify the roles of the different parts of the CNS to make the control of action potential of the cortex used in BCI as stable and robust as possible. However, the current state-of-the-art of BCI research is still with less than the optimal range of robustness to be used for a wide range of clinical and commercial applications which are these systems designed for [2, 5, 6].

The main method of the system design is to develop a controller for a quadcopter to pick up objects and bring it back to the user. Here, we show that it is possible to develop a flexible modular system that can be used for interfacing BCI systems with quadcopter flying robots.

The method of designing BCI systems can be classified in several ways according to different point of views. It can be classified according to the neuroimaging methods used, type of the physiological signals used, the approach of the BCI (exogenous and endogenous), the timing paradigm used (synchronous and asynchronous), feature extraction and feature selection methods used [1-3, 5, 7-9]. Briefly, the major parts of any BCI system will include the following parts: signal acquisition unit, feature extraction unit, feature translation unit and device control interface.

To the best of the author's knowledge, there are only two research papers in the literature describing studies attempted to control quadcopter robots using BCI. The first study has been performed by a research group in University Minnesota (USA) in 2013 [10]. They have used sensorimotor modulation by imagining hand and leg movements to carry out the BCI control. They have tried successfully to control the quadcopter in a large sport field and to control the movement of the quadcopter to fly inside several large circular boundaries (~2 meters in diameter). However, no trials have been carried out to make precise control of the quadcopter flying. The second study has been carried out by a research group in China (Pervasive Computing Group at CCNT Lab, Zhejiang University) in 2012 [11]. They have attempted to control a quadcopter using the sensorimotor modulation by imagining moving a box to left, right, up or down. Again the accuracy of the control is within around 1 meter and nothing has been used to enhance the accuracy of the control beside the use of the BCI itself.

Although the accuracy of the classification of the desired movement of the quadcopter for both studies were around 90% (e.g., desired to move left yielded 90% of correct moving of the quadcopter to the left), nevertheless it could not achieve precise movement of the quadcopter in an error of less than +/- 1 meter. The accuracy of the classification is not enough for timely sensitive control. Even, if highly accurate BCI is achieved in term of correct classification of the desired movement, but without high precision to the time of starting/stopping the movement to that direction, the utility of such system is questionable for a practical BCI system for timely sensitive applications.

METHODOLOGY

The controller algorithm has been implemented in Labview 2013 with AR Drone Toolkit. The developed Labview program's aim is to control the quadcopter and receive the data stream from a variety of sensors' data of the qaudcopter. The received data stream is including: Camera video stream, battery charge percentage, quadcopter's current pitch/roll/yaw degrees and altitude all in real time. Furthermore, the developed software is sending control signals to the quadcopter to change its pitch/roll/yaw degrees to accomplish the desired control and the desired movement of the quadcopter.

The developed software is intended to be interfaced with the Brain Computer Interface algorithm which is employing the user's thinking to control the quadcopter. The BCI algorithm is comprised of the following parts:

1. EEG signal acquisition and filtering software

This is the main hardware that is responsible for picking up the brain's physiological signals. In our system, an Electroencephalograph (EEG) is used, which picks up the electrical signals of the brain. The electrodes of the EEG are distributed on 10-10 EEG placement positions over the scalp of the user (Fig. 1). Each electrode is picking up the electrical activity of the underlying and the nearby neural tissue of the cortex of the brain.

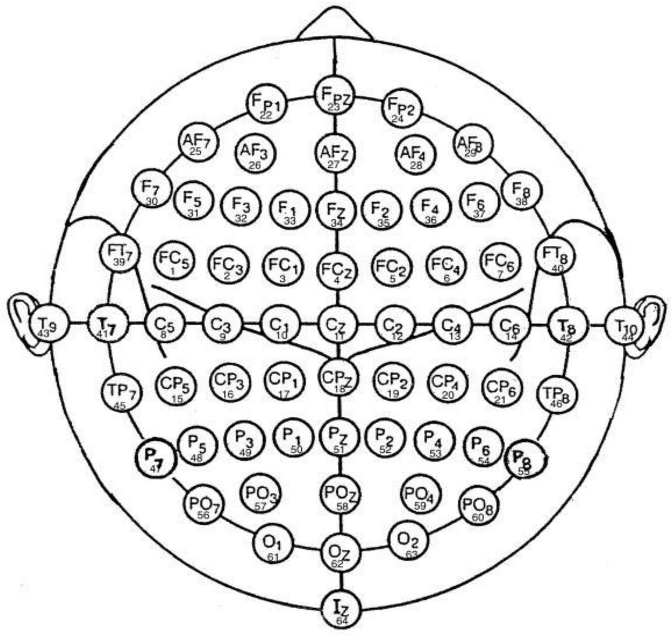

Fig. 1: The channel positions of the 64 EEG electrodes for our proposal

2. BCI feature extraction and translation

a) Feature Extraction

Technically the detection of the movement is based on 3 features that should be extracted from the EEG data. The most significant changed electrode in term of Absolute FFT amplitude is selected to be associated with that specific movement and will be recorded in the profile of that user after repeated the training sessions for several times. Moreover, each electrode is analyzed by each of the 4 standard EEG bands: Delta (1-4 Hz), Theta (4-8 Hz), Alpha (8-12 Hz), Beta (12-25 Hz), High Beta (25-30 Hz), Gamma (30-40 Hz), High Gamma (40-50 Hz), Alpha1 (8-10 Hz), Alpha2 (10-12 Hz), Beta1 (12-15 Hz), Beta2 (15-18 Hz), Beta3 (18-25 Hz), Gamma1 (30-35 Hz), Gamma2 (35-40 Hz) and individual Hz bands (1,2, ... ,49,50 Hz).

The Absolute amplitude with the most significant changes for any of these bands and any of the electrodes for all training trials will be recorded as associated with that movement in the profile database of the user. The Absolute FFT amplitude autospectrum is calculated as the following:

$$X_k = \sum_{k=0}^{N-1} x_n e^{-i2\pi k\frac{n}{N}} \qquad (1)$$

where X_k is the Absolute amplitude of the k individual Hz frequency band, N is half of the sampling rate and n is the sample number.

The FFT absolute amplitude autospectrum is calculated for each epoch. Each epoch contains either 250 ms or 500 ms of EEG samples of a specific EEG channel.

Hence, the three features used for each physiological task are: 1) The most significant EEG channel associated with the physiological task. 2) The associated frequency band. 3) The type of the change of the FFT amplitude autospectrum (decrease or increase in amplitude).

b) Feature Translation

When the training of the classifier is converged, and the user's profile is created, the PC software and the user are ready for doing the experiments. During the experiments, the PC software is analyzing the EEG data and extracting the features, and those features are compared with the profile of the user. Finally, the software is trying to match the EEG features with the most likely physiological task that has been performed by the user.

RESULTS AND DISCUSSION

A modular approach has been followed to develop the quadcopter's controller software. The whole program can be divided into 4 major modules. Each module has its own submodules. The four major modules are shown in Fig. 2, which comprises of : open drone session module, quadcopter control module, Navigation Data read module and Close drone session module.

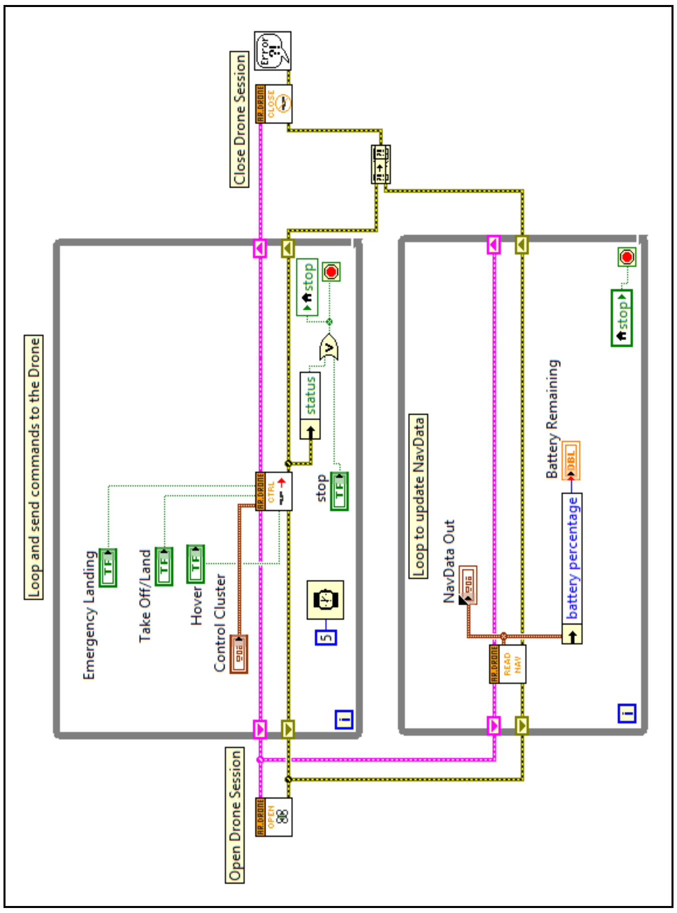

Fig. 2: LabView main program block diagram.

The open drone session module is further comprised of several submodules as shown in Fig. 3 : open control connection module, open NavData connection module, open video connection module and open TCP control port module. The Wifi open session module is responsible for sending the initial setup configuration and handshaking commands through the connected wifi port from the Pc to the quadcopter.

Both TCP and UDP protocols are being used. Particularly, the TCP protocol has been used to send the control commands which are critical and needs this type of robust protocol. While the less critical video stream and sensor data stream are being transferred through the UDP protocol.

The quadcopter drone module's block diagram is shown in Fig. 4. This module is responsible for sending commands to control the following parameters in the drone: Pitch/roll/Yaw degree, vertical speed.

Moreover, it is giving instructions to make the quadcopter performs either Hovering (i.e., staying still in the air, takeoff/Land and Emergency landing. Each of these functions has a specific AT command that should be sent to the drone to execute that specific functions. The AT commands are strings encoded as 8-bit ASCII characters with a 'carriage return' character (CR). The details of these AT commands can be found in the ARDrone Developer Guide [12].

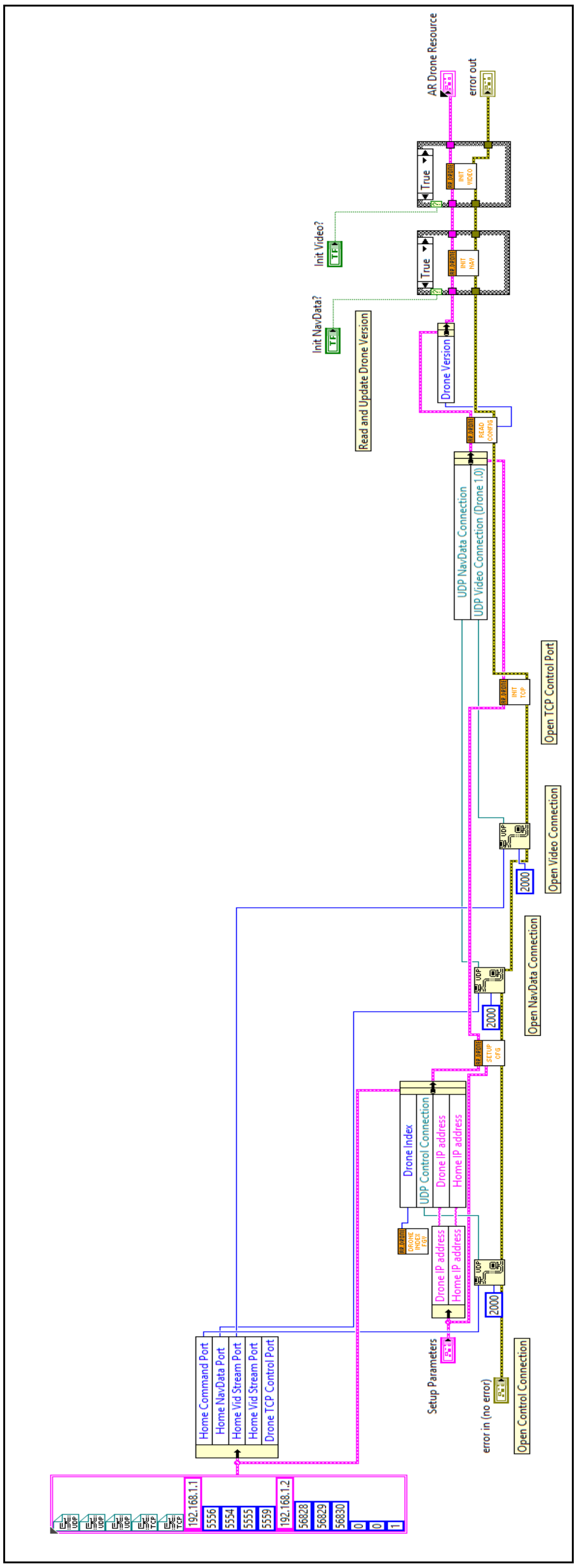

Fig. 3: Open drone session module block diagram.

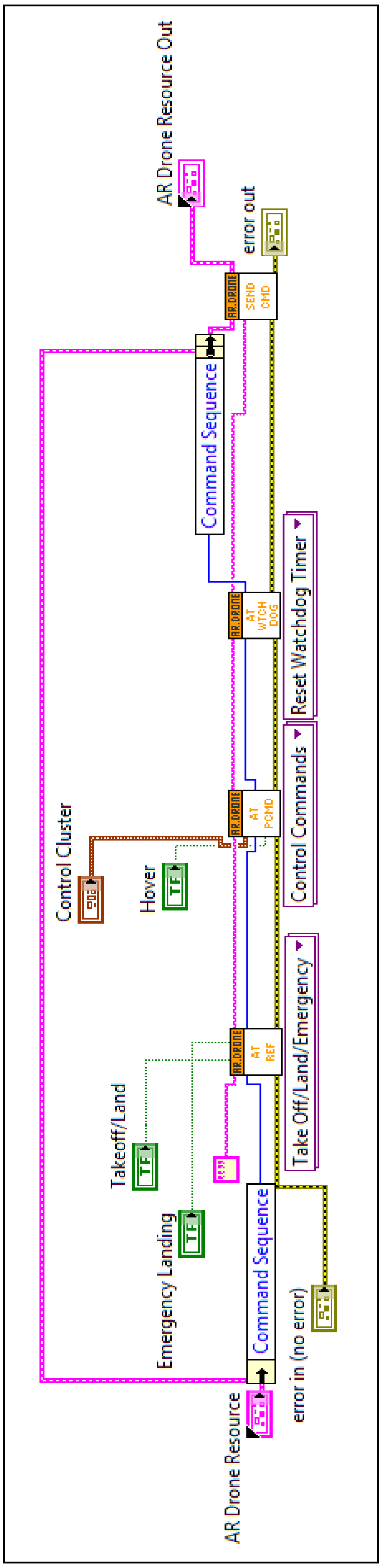

Fig. 4: Quadcopter control module block diagram.

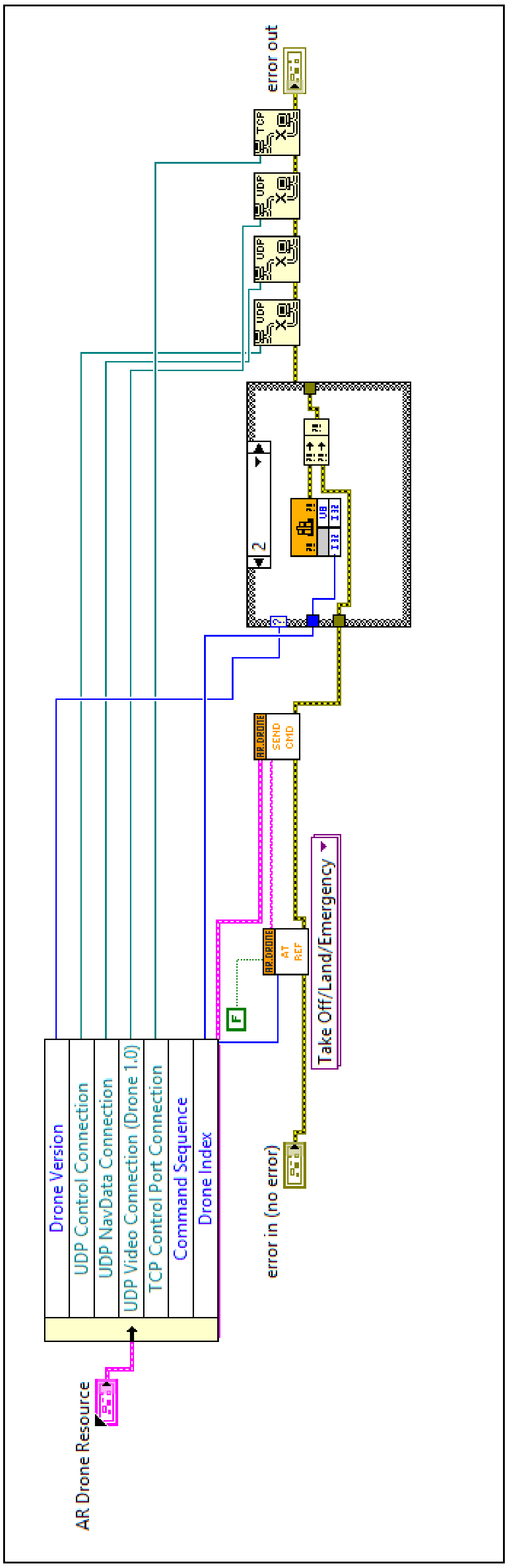

Fig. 5: Close drone session module block diagram.

Fig. 5 shows the close drone session module block diagram. It is responsible for handling any error that can arise in the program. If there is any error, the module will send an emergency Landing AT command to the quadcopter.

Fig. 6 shows the navigation data read module block diagram. It is responsible for receiving a real time data stream of the following data from the quadcopter's sensors: battery charge percentage, the current pitch/roll/yaw degree of the quadcopter and the current altitude.

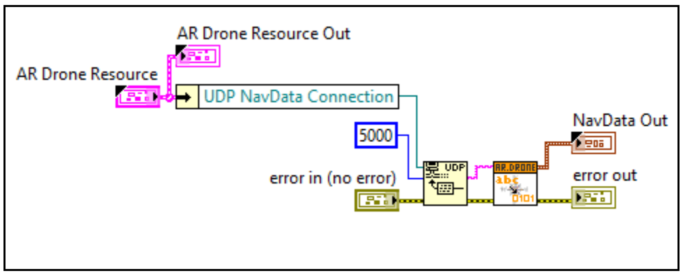

Fig. 6: Navigation data read module block diagram.

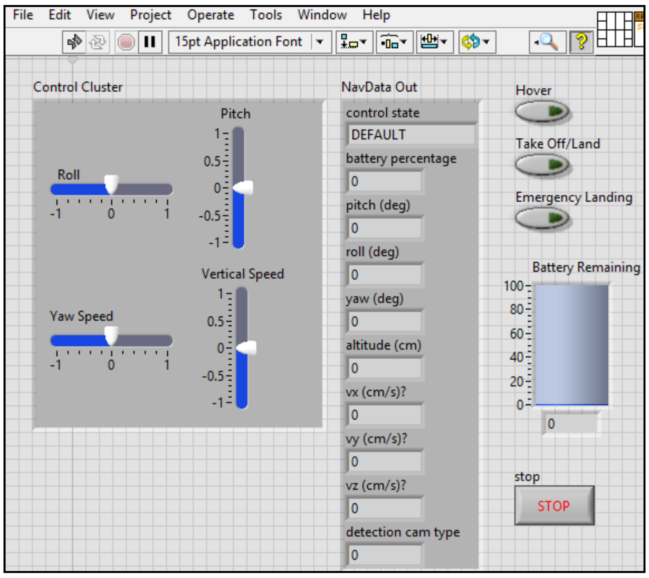

Fig. 7: AR Drone Navigation Software Front Panel.

Fig. 7 shows the software's front panel. The four sliders on the left of the front panel is used to control the pitch/roll/yaw and the vertical speed of the quadcopter. The middle part of the front panels shows the navigation data stream that is received from the quadcopter. And finally on the right side of the front panel there are 3 buttons to send Hover, Take off/Land and Emergency landing AT commands to the quadcopter. Below them is the indicator for the quadcopter's battery charge level.

A real-time parallel process of monitoring the controller sliders to record all the user's interaction with the GUI has been added to the main program block which is shown in Fig. 9. The quadcopter is very sensitive to those sliders in the GUI. It should not be set the sliders on a certain value other than neutral value (i.e., zero) for a more than around 1 second if the quadcopter is flying inside a small room, because of the risk of colliding with the wall or other objects. That's why a dedicated module has been developed to let the sliders return back to neutral value immediately after releasing the left mouse button during changing the sliders' values. Fig. 8 is showing that module which comprises of an Event Structure which is triggered when the left mouse button is released from one of the 4 sliders. The code is being executed whenever the left mouse button is being released in which it will insert a zero value in all the sliders. The local variable "Control Cluster" is the cluster variable that is associated with the 4 sliders' values.

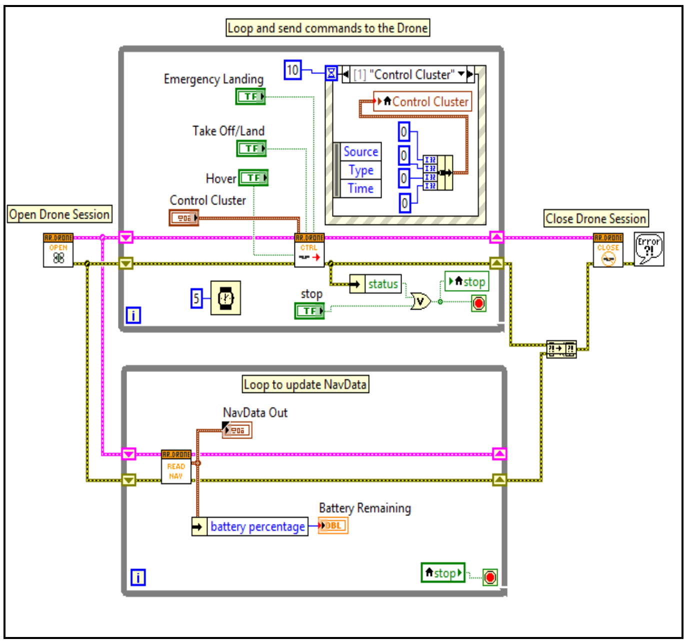

Fig. 8: Real-time parallel process of monitoring the controller sliders added to the main block diagram.

Another Do-while loop is being executed in parallel with the sending commands to the quadcopter loop and the receiving data from the quadcopter loop. This Do-while loop is responsible for recording the state of the 4 sliders and then replaying the negative values of those states in reverse order when the user presses the Replay button. Fig. 9 is showing this record and replay loop. It comprises of 2 inner Do-while loop. The 1st loop is responsible for recording the states of the sliders. The local variable "Control Cluster" is being accumulated in a cluster array and indexed by the loop. When the loop is being stopped by pressing the Replay button, the array is passed to reverse ordering module to reverse the index of the array (i.e., the last element will become the 1st element). Then all the values of the cluster (i.e., the 4 sliders' states) will be multiplied by -1. Thus the cluster array of the sliders' states has been changed into another cluster array which is ready to play it back in a way that will make the quadcopter performs all its movements that has been done in the recording stage, into same movements but in reverse and in their negative sliders' values.

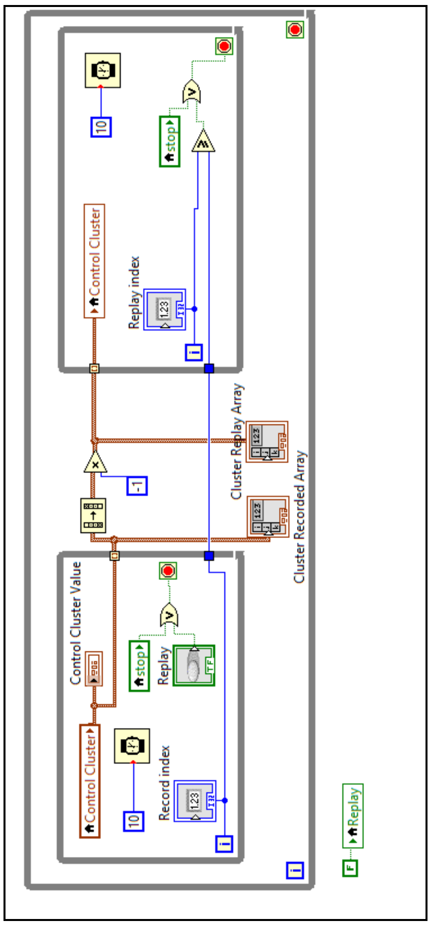

Fig. 9: Record and Replay loops inside their main loop which are responsible of recording the sliders' states and replaying those states in reverse order. This module is executed in parallel with the main module.

For example, if the quadcopter moved 1 second of 0.5 up then 2 second of 0.3 right then 3 second of 0.7 front, the replay of these movements will be 3 seconds of -0.7 front (i.e., back) then 2 seconds of -0.3 right (i.e., left) and finally 1 second of -0.5 up (i.e., down).

This reversed indexed array is passed in the second Do-while loop and it is passed element by element into the Control Cluster local variable. Thus, the sliders are moving on its own in the same way that the user moved them, but in reverse order in term of the time and values. This second loop will be stopped when the iteration number reaches the same iteration number that has been reached during the recording loop. Two indicators will show those iterations in the front panel (replay index and the record index).

CONCLUSION

A software for controlling and receiving real time data stream from a quadcopter flying robot has been demonstrated. This Labview program is capable of controlling and reading all the required data streams. Moreover, it can record all the sliders' states and replaying them back again. The design is a flexible module that can be interfaced with a brain computer interface to perform the intentions of the user to control a quadcopter through his thinking process alone.

Acknowledgment

This research is supported by the Graduate Assistantship Scheme of Universiti Teknologi PETRONAS, and by the Ministry of Education Malaysia under the Higher Institutional Centre of Excellence (HICoE) Scheme awarded to the Centre for Intelligent Signal and Imaging Research (CISIR).

REFERENCES

[1] H. H. Alwasiti, I. Aris, and A. Jantan, "Brain computer interface design and applications: challenges and future," World Applied Sciences Journal, vol. 11, no. 7, pp. 819-825, 2010.
[2] J. Wolpaw and E. W. Wolpaw, Brain-computer interfaces: principles and practice. Oxford University Press, 2012.
[3] L. F. Nicolas-Alonso and J. Gomez-Gil, "Brain computer interfaces, a review," Sensors, vol. 12, no. 2, pp. 1211-1279, 2012.
[4] J. E. Hall, Guyton and Hall textbook of medical physiology e-Book. Elsevier Health Sciences, 2015.
[5] C. Vidaurre and B. Blankertz, "Towards a cure for BCI illiteracy," Brain topography, vol. 23, no. 2, pp. 194-198, 2010.
[6] B. Graimann, B. Z. Allison, and G. Pfurtscheller, Brain-computer interfaces: Revolutionizing human-computer interaction. Springer Science & Business Media, 2010.
[7] Walker, M. Deisenroth, and A. Faisal, "Deep convolutional neural networks for brain computer interface using motor imagery," MSc. Computing Science, Imperial College London, 2015.
[8] R. Kus, D. Valbuena, J. Zygierewicz, T. Malechka, A. Graeser, and P. Durka, "Asynchronous BCI based on motor imagery with automated calibration and neurofeedback training," IEEE Transactions on Neural Systems and Rehabilitation Engineering, vol. 20, no. 6, pp. 823-835, 2012.
[9] V. Kaiser et al., "Cortical effects of user training in a motor imagery based brain–computer interface measured by fNIRS and EEG," Neuroimage, vol. 85, pp. 432-444, 2014.
[10] K. LaFleur, K. Cassady, A. Doud, K. Shades, E. Rogin, and B. He, "Quadcopter control in three-dimensional space using a noninvasive motor imagery-based brain computer interface," Journal of neural engineering, vol. 10, no. 4, p. 046003, 2013.
[11] Y. Yu et al., "FlyingBuddy2: a brain-controlled assistant for the handicapped," in Proceedings of the 2012 ACM Conference on Ubiquitous Computing, 2012: ACM, pp. 669-670.
[12] P. Eline and F. D'Haeyer, "ARDrone Developer Guide," www.msh-tools.com/ardrone/ARDrone_Developer_Guide.pdf, 2014.