Industrial Implementation of a Fuzzy Logic Controller for Brushless DC Motor Drives using the PicoMotion Control Framework

Sevil Ahmed, Andon Topalov, Nikolay Dimitrov
Control Systems Department, Technical University of Sofia,
Plovdiv branch
25, Tsanko Dyustabanov St., 4000 Plovdiv, Bulgaria
e-mail: {sevil.ahmed; topalov}@tu-plovdiv.bg
dd.nikolay@gmail.com

Eugene Bonev
PicoMotion Inc.,
Santa Clara, CA, USA
e-mail: eugenebonev@gmail.com

***Abstract*- Modern industry requires an improved motor performance and the permanent magnet brushless DC (BLDC) motors are especially appropriate for applications that require a high level of accuracy and performance. The advanced control strategies when applied into BLDC motor drive systems may further enhance their performance. However, these methods are relatively complex and computationally intensive, which may hamper the implementation. Fuzzy logic control algorithm has been developed in this investigation and embedded into a commercially available motion control system for the position control of BLDC motors. BLDC motor drives offered by PicoMotion Inc. and Motion Control Framework software platform have been selected for the experiments due to their compactness, flexibility and the open hardware and software concept they have been created upon. The obtained comparative results with respect to the built in PI control law have confirmed that the implementation of more advanced control strategies on inexpensive hardware and software platforms, nowadays available on the market for the control of BLDC motors, can be practically feasible and can enhance their performance.**

Keywords- fuzzy logic control; industrial implementation; BLDC motor drives

I. INTRODUCTION

During the past ten years, the permanent magnet brushless motor market has grown much faster than the other small motor markets. Permanent magnet brushless DC (BLDC) motors are very appropriate for applications where a high level of accuracy and performance are required. They are widely used in robotics, precision machine tools, automotive electronics, military applications, aerospace, industrial process control, household appliances and office automation.

The BLDC motor control system is a nonlinear and multivariable coupling system. Its performance can be further improved by: i) optimizing the motor design, ii) optimizing the control of power electronic devices and iii) implementing advanced control strategies. It is to be noted that the traditional "proportional plus integral plus derivative" (PID) control algorithm commonly built in the industrial motion control systems is very intuitive and easy to implement. However, the demand for high precision servo-control systems characterized with the presence of nonlinear dynamics requires implementation of more advanced control strategies.

Project 161ПР0009-19 by Research and Development Sector of Technical University of Sofia, branch Plovdiv and within the Ministry of Education and Science of Bulgaria Research Fund Project FNI I 02/6

Nonlinear control methods based on the modern control theory and intelligent control, have established the foundation for high-quality dynamic and stable performance and are highly recommendable for use in the BLDC motor control systems. Fuzzy control, neural network-based control, variable structure control, robust control, adaptive control and other advanced control strategies have been investigated for the control of BLDC motor drives [1-7]. The problem with the advanced control methods is that they are relatively complex and computationally intensive, which may hamper their implementation in the industrial BLDC motor drive systems [8]. Most of the recently published results in this field are obtained by simulations or with a specialized hardware for rapid prototyping [9, 10].

Fuzzy logic control (FLC) has been embedded in this investigation into an inexpensive commercially available BLDC motor control system and its performance has been evaluated. Dual axis motion control modules offered by PicoMotion Inc. and the Motion Control Framework (MCF) software platform, developed by one of the authors of this investigation, have been chosen for the experiments due to their compactness, flexibility and open hardware and software concept they have been built upon. FLC can improve the performance of the BLDC motor drive system on torque-ripple minimization, dynamic and steady state speed response and increase its robustness with respect to disturbances and unmodeled dynamics. Since the selected PicoMotion control framework is dedicated to the design of control applications in the field of robotics, it will facilitate the next planned step within this investigation aiming to evaluate the performance of the developed FLC algorithm when applied to the synchronous joints control of a Milara Inc. ATM 105 wafer handling robot manipulator, formerly a Brooks product.

II. MOTION CONTROL FRAMEWORK

The structure of the Motion Control Framework is presented on Fig. 1. The MCF is a main part of specialized embedded controller and incorporates basic features for BLDC control in robotics.

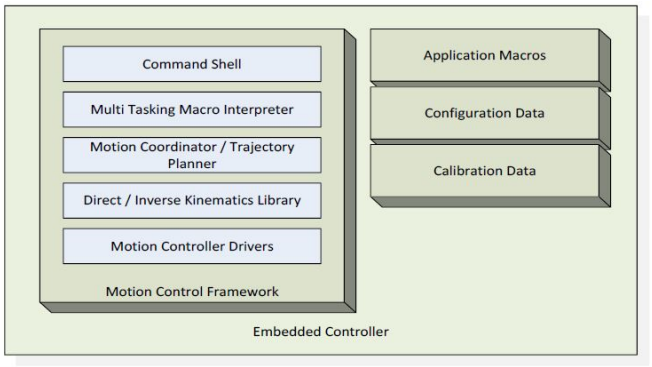

Figure 1. Motion Control Framework structure.

A. Motion Control Language

The Motion Control Language (MCL) is a software package that provides domain specific programming language and multi-tasking, real-time interpreter.

The MCL executable includes integration of motion controllers and I/O modules, implementation of various robot kinematics, support for complex trajectory motions as well as monitoring of the system integrity. The executable also includes real-time multi-tasking script interpreter. It allows the design of application specific scripts which can be used to coordinate the sequencing of robot motions and implement custom host communication protocols.

B. Command Shell

The Command Shell (see Fig.1) acts as user interface for the Motion Control Framework. The Command Shell "listens" to several input streams: the console MCF is started on (over serial interface) a Teach Pendant interface (auxiliary serial interface) and on a TCP port (raw Sockets interface).

C. Multitasking Macro Interpreter

1) Real-Time Tasks

The real-time tasks are designed to ensure the deterministic behavior of the controlled system. The MCF executes several real-time tasks periodically on a fixed time interval. The list of real-time tasks includes:

- Reading hardware state – position, status, digital inputs
- Checking for critical error conditions – evaluating interlocks,
- Executing user-defined critical condition,
- Generating velocity profile – define the motion trajectory and coordination between motors,
- Executing user defined "Supervisor" macro – implement PLC like functionality,
- Updating the digital outputs,
- Streaming the new trajectory points to the motion controllers.

The interval between the executions of the real-time tasks is referred to as time slice. The length of the time slice depends on the complexity of the tasks which in turn depends on the number of motors that need to be controlled as well as the length of the user defined operations. Typically, the time slice varies from 4 to 10 ms. It can be configured depending on the needs of the application.

2) Non Real-Time Tasks

The non-real-time tasks have no deterministic behavior. These tasks are responsible for the sequencing of the individual motions or the communications with the host computer. The non-real-time task belongs to one of the following two categories:

- Foreground task – handles the host communications and the operation of the command shell.

- Background task – it executes user defined procedures in "background". These tasks are started by the foreground tasks in response to requests for execution of a sequence of motions or activities.

D. Motion Control Features

The motion control features are the main benefit of using the Motion Control Framework. The software includes rich set of commands and features that allow the design of complex applications with little or no concern for the low level details related to the setup, coordination and error handling associated with the motion controllers or the motors being controlled.

E. Application Structure

Each MCF application consists of minimum Configuration and Macro files.

1) Configuration File

The configuration file contains definitions of the objects that need to be created for the needs of the specific application. Each object is instance of one of the following hardware abstraction classes: Board, Input, Output, IO Module, Robot. In addition, the configuration file contains global parameter settings affecting the operation of the whole system.

2) Application Macros

The application macro file contains the business logic of each application as well as application data definitions. The names of the macros and their parameters can be regarded as the API exposed to the host computer. The macro file can include a Supervisor macro that implement PLC like functionality. It is executed in its entirety every time slice.

A dedicated macro can be assigned to act as run-time exception handler. This macro is invoked automatically at run time if a critical error is detected in any of the motion controllers.

3) Procedure and Macro Calls in MCF

The MCL supports two types of subroutines – procedures and macros. Their only difference is their visibility from the command line interpreter and their execution context. The procedures are not visible directly – they can still be invoked with a prefix (dot) in front of their name. If a procedure is invoked this way its execution takes place in „background".

The macros are directly visible from the command line. When invoked the execution takes place in the „foreground".

The definition of a procedure starts with the keyword PROC followed by the procedure name and the list of expected parameters. The definition of a macro starts with the keyword MACRO followed by the macro name and the list of expected parameters.

The procedure or macro should be followed by RETURN statement before any new procedure definition is allowed.

Examples:

```
PROC ExampleProcedureName  Par1:NN, Par2
RETURN Par1 * Par2

MACRO TST Par1:NN
RETURN
```

The parameters defined after the procedure's names are comma delimited list of identifier and type code separated by column sign. The format of the parameter definition is

Identifier: TypeCode

The type code should be one of the following:

NN – Necessary number
ON – Optional number
NA – Necessary axis
OA – Optional axis
NC – Necessary character
OC – Optional character

F. Hardware Abstraction Classes

The hardware classes abstract the parameters and the functionality of the controlled Axes, Inputs, Outputs, and Boards. The diagram below shows their relationships (Fig. 2).

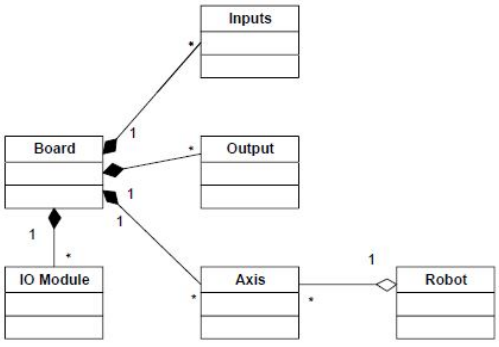

Figure 2. Hardware abstraction classes.

The Board class abstracts the union of servo axes, digital inputs, outputs and IO modules sharing the same communication channel. In the case of a centralized motion controller all of the above hardware interfaces are implemented on a single PCB – hence the name is Board. In case of a distributed motion control the Board represents the communication bus (e.g. serial port) connecting the individual motion control modules. The Board class encapsulates information such as base address (for a centralized controller) or communication port number (for a distributed controller). In addition, it also includes communication parameters (baud rate, protocol, etc.).

III. FUZZY LOGIC CONTROLLER DESIGN IN MCF

The basic structure of FLC is adapted to the concept for positioning control of BLDC motor drive (Fig. 3).

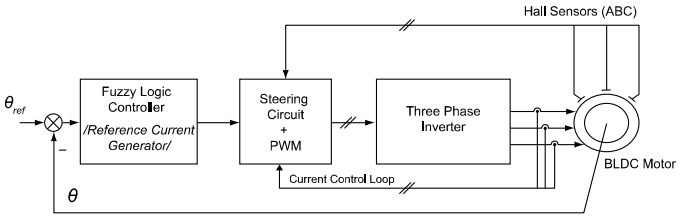

Figure 3. Fuzzy logic position control of BLDC Motor – basic structure.

The FLC replaces the conventional PI(D) controller in this widely used structure of positioning control of BLDC motors in robotics applications. Implemented FLC attempts to minimize the positioning error $e(k) = \theta_{ref}(k) - \theta(k)$, which is calculated as difference between the desired position $\theta_{ref}(k)$ and the actual position $\theta(k)$ measured by incremental encoders at each time instant *k*. The output of the controller generates corrective action, which subsequently is transformed into PWM signal by Steering Circuit and PWM block in the structure.

The implemented FLC is designed with two inputs – the positioning error $e(k) = \theta_{ref}(k) - \theta(k)$ and its increment $\Delta e(k)$. Thus, the controller has the classic design, which is shown in Fig. 4.

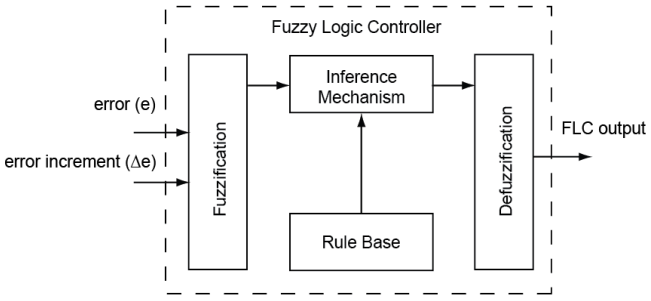

Figure 4. Block diagram of the fuzzy controller.

The structure of the fuzzy logic controller is adapted to MCF using following basic procedures: Fuzzify, RuleBase and Defuzzify. Each of them performs a particular stage of the fuzzy logic conception (Fig. 4).

A. Fuzzification Procedure

The fuzzification procedure is the first stage of fuzzy logic control strategy. According the nature of the procedure it accepts two arguments: the number of the membership functions in the fuzzy set (count) and the input signal, which is going to be fuzzified (inp).

```
PROC Fuzzify inp:NN, count:NN
      VAR i, j
      i = 0
      j = 0
      for_loop:
      fuzzy_array[j] = Triangle array[i], array[i+1], array[i+2], inp
        j = j+1
        i = i+3
        if j < count
                goto for_loop
        endif
RETURN
```

1) Membership functions

There are two alternative ways to represent a membership function: continuous or discrete. Here, a continuous fuzzy set A is defined using a continuous membership function $\mu_A(x)$. The triangle membership function is a piecewise linear, continuous function, controlled by three parameters $\{a, b, c\}$.

$$\mu_A(x) = \begin{cases} \dfrac{x-a}{b-a} & , a \le x \le b \\ 1 & , x = b \\ \dfrac{c-x}{c-b} & , b \le x \le c \end{cases} \qquad (1)$$

The parameters $a \le b \le c$ define three breakpoints, designated as follows: left footpoint (a), midpoint (b) and right footpoint (c). Fig. 5 illustrates a triangle membership function.

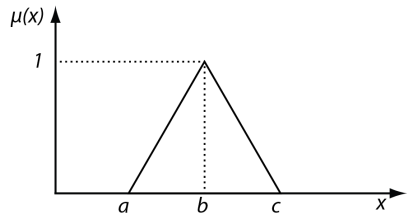

Figure 5. Triangle membership function.

The "triangle" procedure is created according (1) and MCL coding rules. All of the procedure parameters are defined to be NN (see Section 2.E). This procedure returns the membership degree for the input signal x.

```
PROC Triangle a:NN, b:NN, c:NN, x:NN
    if (x >= a) && (x <= b)
      m =((x - a) / (b - a))
    endif
    if (x == b)
      m = 1
    endif
    if (x >= b) && (x <= c)
      m = ((c - x) / (c - b))
    endif
    if (x < a || x > c)
      m = 0
    endif
RETURN m*10000
```

B. The Rule Base Procedure

The rule base of the implemented FLC converts the control logic into an appropriate for programming "if-then" conception. It is a necessary step to fuzzy inference process, which combines membership functions with the defined rules to acquire the fuzzy output. Linguistic variables and their values are usually used to create a 2D fuzzy relation matrix (Table I), which is based on Mamdani fuzzy inference model.

TABLE I. FUZZY RULE BASE

e / Δe	**NB**	**NS**	**Z**	**PS**	**PB**
NB	NB^0	NB^5	NB^{10}	NS^{15}	Z^{20}
NS	NB^1	NB^6	NS^{11}	Z^{16}	PS^{21}
Z	NB^2	NS^7	Z^{12}	PS^{17}	PB^{22}
PS	NS^3	Z^8	PS^{13}	PB^{18}	PB^{23}
PB	Z^4	PS^9	PB^{14}	PB^{19}	PB^{24}

TABLE II. IMPLEMENTING FUZZY RULES IN MCL

e / Δe	**1**	**2**	**3**	**4**	**5**
1	1^0	1^5	1^{10}	2^{15}	3^{20}
2	1^1	1^6	2^{11}	3^{16}	4^{21}
3	1^2	2^7	3^{12}	4^{17}	5^{22}
4	2^3	3^8	4^{13}	5^{18}	5^{23}
5	3^4	4^9	5^{14}	5^{19}	5^{24}

Some notations should be given, in order to define this type of fuzzy rules in MCF. It is convenient to represent the rule base from Table I by one-dimensional array. Here comes the notational mapping of the linguistic values for Negative Big (NB), Negative Small (NS), Zero (Z), Positive Small (PS) and Positive Big (PB) position error (e) or its increment (Δe) using integer values like 1 for NB, 2 for NS, 3 for Z, 4 for PS and 5 for PB (Table II). Superscripts in both tables correspond to the index of the element in the designed rule base.

Thus, the array, which represents the rule base in Tables I and II, must be given by following manner:

```
RuleBaseMatrix[0] = 1
RuleBaseMatrix[1] = 1
RuleBaseMatrix[2] = 1
RuleBaseMatrix[3] = 2
.....
RuleBaseMatrix[24] = 5
```

Created RuleBase procedure accepts two parameters -inputs of the FLC.

```
PROC RuleBase inp1:NN, inp2:NN
```

According Fig. 3 and 4 the inputs are e and Δe. They are derived as: $e(k) = \theta_{ref}(k) - \theta(k)$ and $\Delta e(k) = e(k) - e(k-1)$, where the position $\theta(k)$ is ensured by the MCF "enc" command. It returns/reads the current position in encoder counts for motor associated with axis X.

```
Theta = Enc X
```

The RuleBase uses the Fuzzify procedure and returns the activated rules according Min-Max inference mechanism. Result is an array with nine elements – the codes of output membership functions (1, 2, 3, 4 or 5).

C. *Defuzzification Procedure*

This stage produces a crisp output value of the FLC. Here the Centroid defuzzification method is implemented. The procedure has three parameters – inputs (inp1 and inp2) of the FLC and again the number of the membership functions in the fuzzy set (count).

```
PROC Defuzzify inp1:NN, inp2:NN, count:NN
```

It returns the defuzzified output of the FLC. Afterward, this signal is used to determine the PWM duty cycle value which is going to be applied to the motor in order to ensure the desired positioning performance (Fig. 3). In MCF it is performed by following command:

```
Motor X = u
```

where X the axis on which motor acts. The "Motor" command applies a power to the BLDC motor using the PWM signal with duty cycle. It varies between -32 767 and +32 767. Limit values correspond to 100% PWM power to the motor, which results in rotation in clockwise or counterclockwise direction. The value determines the speed of motion.

IV. EXPERIMENTAL TESTS

A. *BLDC Control Testbed*

The BLDC motor control testbed that has been used for the comparative experiments is shown on Fig. 6.

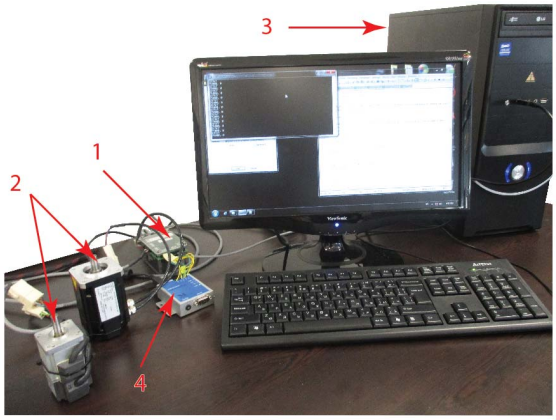

Figure 6. BLDC control testbed. 1) PicoMotion cotrol module, 2) two BLDC motors, 3) computer, 4) USB to RS422 adapter.

Its main components are the dual-axis PicoMotion control module, two permanent magnet BLDC motors and a PC with running MCF. Since the interface between the PicoMotion control module and the computer has to be implemented via RS422 communication protocol, an USB to RS422 adapter has been additionally used.

1) *PicoMotion Dual-Axis Motion Controller*

The dual-axis motion control module PMC201/PMC202 (Fig.7) is a development tool for the NXP MC56F8257 digital signal controllers/processors. Well suited for motor control applications, it can be combined and used with several peripherals or be operated as a standalone tool. It can also be used as the main control board with an appropriate motor control board. Main features of the controller could be seen in [11].

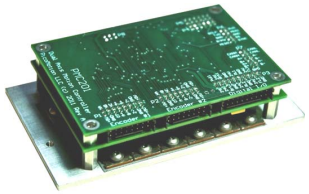

Figure 7. PicoMotion dual-axis motion controller PMC201/PMC202.

2) *Panasonic BLDC motor*

A 100W Panasonic permanent magnet BLDC servo motor has been used for the conducted experiments. It has been selected for testing since motors from this type are currently implemented within the joints actuators of the Milara Inc. AXM 100 series wafer handling atmospheric robots. The next planned research step will include performance evaluation of the developed FLC algorithm for PicoMotion BLDC motor drives when the later are applied to the synchronous joints control of an ATM 105 manipulator. The motor specifications are presented in Table III.

TABLE III. PANASONIC BLDC MOTOR PARAMETERS

Power (kW)	0.05
Input Power Supply (V)	200
Rated Torque (N·m)	0.16
Peak Torque (N·m)	0.48
Rated rotational speed (rpm)	3000
Moment of inertia of rotor ($\times 10^{-4}$kg·m^2)	0.025
Rated Current (A (rms))	1
Encoder	Incremental

B. *Experimental Results*

This section presents the main results obtained from the experimental performance evaluation tests of the developed FLC algorithm for PicoMotion BLDC motor drives. The experiments have been performed with a time slice T_s=0.06 sec and changing reference signal for the position. The fuzzification of the inputs has been implemented via fuzzy sets with five triangular membership functions (Fig. 8). The same functions have been used for the consequent part of the if-then rules too. Before applying the fuzzifying procedure each input has been normalized within the interval of [-1 1]. Whereas, the final output from the FLC has been then multiplied by an appropriate gain in order to apply an adequate control signal value to the BLDC motor.

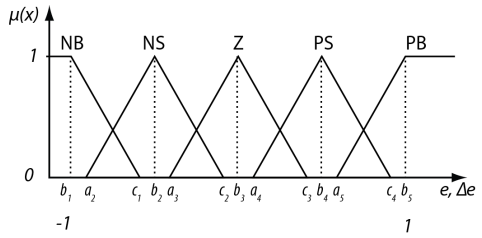

Figure 8. Membership functions of the error and its increment for the implemented FLC

The obtained results from the conducted comparative tests (Fig. 9) with conventional "proportional plus integral" (PI) controllers (with two selected sets of parameters presented on Table IV) show better performance of the system, when it is working with the implemented FLC law.

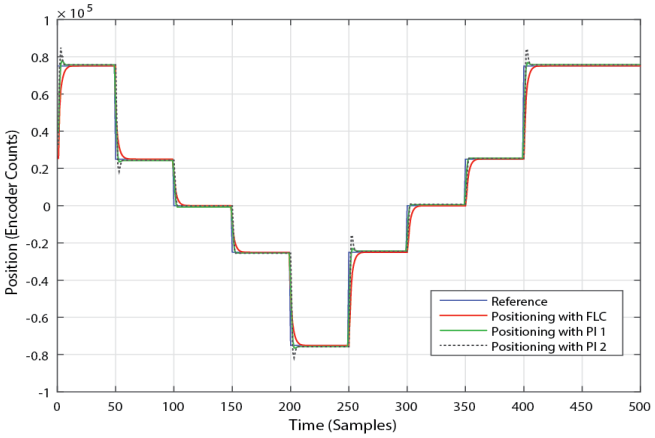

Figure 9. Comparative positioning results
(2 500 encoder counts = 1 turn).

It can be observed that, due to the BLDC motor control system nonlinearity, the performance of the PI controllers depends on the reference increment. Bigger changes result in overshooting. Hence, the performance of FLC ensures constant performance.

TABLE IV. PI CONTROLLER PARAMETERS

Controller	k_p	k_i
PI 1	0.75	2.25
PI 2	0.75	5.5

V. FUTURE WORK

Future experiments will include performance tests carried on an ATM 105 wafer handling atmospheric robot manipulator equipped with PicoMotion servo drives, with built in fuzzy logic control algorithm, for the synchronous joints position control (Fig. 10). The designed FLC algorithm will be compared with conventional PID and PI control laws where a special emphasize will be putted on the positioning accuracy and presence of vibrations.

Figure 10. The ATM 105 atmospheric manipulator

VI. CONCLUSIONS

A FLC algorithm has been developed and built into a commercially available motion control system for BLDC motors. The results obtained in copmarison with the built in PI control law have shown that the implementation of advanced control strategies on inexpensive hardware and software platforms for control of BLDC motors, may be practically feasible and can enhance their performance.

ACKNOWLEDGMENT

The authors gratefully acknowledge the financial support provided within the Project 161ПР0009-19 by the Research and Development Sector of the Technical University of Sofia and within the Ministry of Education and Science of Bulgaria Research Fund Project FNI I 02/6.

REFERENCES

[1] N. J. Sujatha, M. Saravanan, "Comparative study of fuzzy logic controllers for BLDC motor drive," ARPN Journal of Engineering and Applied Sciences, vol. 10, no. 9, May 2015, pp. 4167-4175.

[2] C. K. Lee, N. M. Kwok, "A variable structure controller with adaptive switching surfaces for brushless DC motor," Proc. of American Control Conference, USA, vol. 1, 1995, pp. 1033-1034.

[3] C. L. Xia, X. J. Yang, T. N. Shi, "Robust speed controller design for brushless motor drive," Advanced Technology of Electrical Engineering and Energy, vol. 21, issue 3, 2002, pp. 5-8.

[4] A. Rubaaai, D. Ricketts, N. D. Kankam, "Development and implementation of an adaptive fuzzy-neural-network controller for brushless drives," IEEE Transactions on Industry Applications, vol. 38, no. 2, March/April 2002, pp. 441-447.

[5] Z. Q. Li, C. L. Xia, "Speed control of brushless DC motor based on CMAC and PID controller," IEEE Proc. of the World Congress on Intelligent Control and Automation, China, 2006, pp. 6318-6322.

[6] D. K. Panicker, M. R. Mol, "Hybrid PI-fuzzy controller for brushless DC motor speed control," IOSR Journal of Electrical and Electronics Engineering (IOSR-JEEE), vol.8, issue 6, Nov.- Dec. 2013, pp.33-43.

[7] Y. Tian, T. N. Shi, C. L. Xia, "Sensorless position control using adaptive wavelet neural network for PM BLDCM," Proc. of IEEE International Symposium on Industrial Electronics, Spain, 2007, pp. 2848-2852.

[8] C. L. Xia, Permanent Magnet Brushless DC Motor Drives and Controls. John Wiley & Sons Singapore Pte. Ltd., 2012, 277 p.

[9] A. R. Anjali, "Control of three phase BLDC motor using fuzzy logic controller," Int. Journal of Engineering Research and Technology (IJERT), vol. 2 (7), July, 2013, pp. 689-693.

[10] T. Ch. Siong, B. Ismail, S. F. Siraj, M. F. Mohammed and M. F. N. Tajuddin, "Implementation of fuzzy logic controller for permanent magnet brushless DC motor drives," Proc. of IEEE Int. Conf. on Power and Energy (PECon2010), Nov.-Dec., Malayia, 2010, pp. 462-467.

[11] MCL5 User Manual. PicoMotion.