Technical Sciences, 2020, 23(3), 209–220

UNIWERSYTET WARMIŃSKO-MAZURSKI W OLSZTYNIE

Quarterly peer-reviewed scientific journal
ISSN 1505-4675
e-ISSN 2083-4527
TECHNICAL SCIENCES
Homepage: www.uwm.edu.pl/techsci/

DOI: https://doi.org/10.31648/ts.5696

A CONTROLLER FOR BRUSHLESS DIRECT CURRENT ELECTRIC MOTORS PART 2: SOFTWARE

Zenon Syroka

ORCID: 0000-0003-3318-8495
Technical Science Department
University of Warmia and Mazury

Received 24 July 2020, accepted 1 December 2020, available online 3 December 2020.

K e y w o r d s: digital control, motor controller electric and hybrid vehicles, brushless motors, microcontroller.

A b s t r a c t

A universal controller for brushless direct current (BLDC) motors was designed in the presented article. The system is controlled from the user console where operating parameters are set by the user. Signals are transmitted by cables to microcontrollers which control and monitor electric motors. Microprocessors communicate via a data bus. The controller contains the user console module and the motor control module. The user console module generates commands, and motors are controlled and monitored by the control module. Motor control modules operate independently, and each brushless motor has a dedicated control module. Brushless motors can be controlled in bipolar or unipolar mode. The control method is selected by the operator. The user console and motor controllers communicate via the I²C bus.

Correspondence: Zenon Syroka, Katedra Elektrotechniki, Energetyki, Elektroniki i Automatyki, Wydział Nauk Technicznych, Uniwersytet Warmińsko-Mazurski, ul. Oczapowskiego 11, 10-719 Olsztyn, e-mail: zenon.syroka@uwm.edu.pl, syrokaz@onet.eu

Introduction

The article consists of two parts. The first part describes the electronic and electrical design of the motor controller, and the second part presents the applied software. The described controller for brushless motors has been designed and patented by Z. SYROKA and K. KRAJEWSKI (patent No. P431380, filing date: 4 October 2019).

Industrial machines, vehicles, power tools and household appliances have motor drive elements that power mechanical components. The popularity of electric cars has been increasing in the 21st century. Batteries and motors are the key components of electric vehicles. Batteries are positioned at the bottom of the chassis to lower the vehicle's center of gravity. The motor is located in the front or at the back of the vehicle (Tesla Model S). The motors in electric cars are powered by direct (DC) or alternating current (AC). Brushless motors are one type of DC motors. Rapid advances in electrical and power engineering have introduced novel components to brushless motors, including thyristors, diodes and MOSFET transistors. Electronic subassemblies can turn power supply on and off hundreds or even thousands of times within one second. In electric bicycles, a brushless motor is placed inside the wheel hub. Power is transferred to the wheel, which reduces the physical effort associated with pedaling by 50%.

This article discusses a controller for brushless DC motors that are mounted inside the wheels of land vehicles. The motor is controlled by the user via the user console. The user can change motor speed, driving direction and deploy the electric braking system. The controller controls the steering system, the drivetrain and the braking system. Brushless DC motors can be controlled in unipolar or bipolar mode.

Microcontroller software

The controller contains three microcontrollers. Each microcontroller requires dedicated software for performing its functions. The described controller features Atmega328 microcontrollers which were programmed in the Arduino environment. The Arduino tool resembles a simple text editor, and it is divided into three main sections: toolbar, message area, and code editor. The toolbar at the top features command buttons. The code editor is where the sketches (programs) are written. The message area displays information about the status of the current sketch and possible errors.

The programs that control brushless motors have similar codes. The code for the master microcontroller is related to the user panel, and it sends commands to the microcontrollers that drive brushless motors.

Code for the master microcontroller

The master microcontroller receives and processes data from the user panel and sends commands to slave microcontrollers that drive brushless motors. The master microcontroller controls motor speed and enables the vehicle to move in different directions. The user operates push-buttons and a potentiometer on the user panel to send information about driving speed and direction to the master microcontroller. The data received by the master microcontroller is processed by algorithms and transmitted by the I²C bus to slave microcontrollers.

Microcontroller pins are identified, libraries are included and variables are added in the first step of the program (Fig. 1).

```
3  #define przod 4
4  #define tyl 5
5  #define prawo 6
6  #define lewo 7
7  #define hamulec 8
8  #define potencjometr A0
9  #include <Wire.h>
10 int predkoscAktualna, predkoscPoprzednia=0;
```

Fig. 1. Code for the master microcontroller – definitions

Command *#define* gives a name to a constant value. For example, value 4 is preceded by the name *right* because the SW1 button is connected to the pin corresponding to this value. Command *#include <Wire.h>* includes an external library in the code. The Wire library enables devices to communicate via the I²C bus. Command *int currentSpeed previousSpeed* adds two values that store information about motor speed. The setup function is described in the next step of the program (Fig. 2).

```
46 void setup() {
47   pinMode(przod,INPUT_PULLUP);
48   pinMode(tyl,INPUT_PULLUP);
49   pinMode(prawo,INPUT_PULLUP);
50   pinMode(lewo,INPUT_PULLUP);
51   pinMode(hamulec,INPUT_PULLUP);
52   Wire.begin();
53   WyborSterowania();
54 }
```

Fig. 2. Code for the master microcontroller – setup function

The *void setup* function is called when the microcontroller is powered up. The function code with the instructions to be executed is placed inside braces. Command *pinMode (forward, INPUT_PULLUP)* informs the microcontroller that the pin connected to the forward drive button is an input pin that receives data. Successive commands carry the respective information for the remaining microcontroller pins. Command *Wire.begin* connects the master microcontroller to the I²C bus. Command *ControlMode* calls a function which describes the motor control mode selected by the user. The control mode is selected by the operator with the use of left or right turn switch. The left turn switch activates the unipolar mode, and the right turn switch activates the bipolar mode. Function codes are presented in Figure 3. The *while* command is a loop that checks the expression inside the parentheses. When the expression is true, the function inside the braces below the command is called. The absence of code inside braces denotes a void loop that will stop the program. The program waits until the expression inside the parentheses becomes false, in this case – until the right or left turn button is

```
16 void WyborSterowania(){
17 while((digitalRead(prawo)==HIGH)&&(digitalRead(lewo)==HIGH))
18   {  }
19   if(digitalRead(prawo)==LOW){
20   Wire.beginTransmission(7);
21   Wire.write("b ");
22   Wire.endTransmission();
23
24   Wire.beginTransmission(8);
25   Wire.write("b ");
26   Wire.endTransmission();
27   while(digitalRead(prawo)==HIGH)
28     {       }   }
29   if(digitalRead(lewo)==LOW){
30   Wire.beginTransmission(7);
31   Wire.write("u ");
32   Wire.endTransmission();
33
34   Wire.beginTransmission(8);
35   Wire.write("u ");
36   Wire.endTransmission();
37   while(digitalRead(lewo)==HIGH)
38   {       }    }
39 }
```

Fig. 3. Code for the master microcontroller – selection of motor control mode

pressed. Commands *if (digitalRead(right)==LOW)* and *if (digitalRead(left)==LOW)* are executed when the corresponding expressions are true.

For example, when the left turn button is pressed, the second *if* statement becomes true and information is transmitted to slave microcontrollers. Command *Wire.beginTransmission(7)* initiates communication with device 7 connected to the I²C bus. Command *Wire.write("bipolar")* sends data to an external device. In this case, the transmitted information is the word *bipolar* which denotes the selected motor control mode. Command *Wire.endTransmission()* ends the data transfer. Each command is repeated twice because data are sent to two microcontrollers – 7 and 8. The *while* loop after the *Communication* command makes the program wait until the user releases the button.

The code defines the direction of motor rotation. The *if* statement is called when the user presses the forward drive button (Fig. 4). The *Communication* function is called to send information to external microcontrollers. In the parentheses, 7 is the address of the receiving device, and 1 is the direction of motor rotation. The code for reverse drive (Fig. 5) and braking (Fig. 6) is similar.

```
80   if(digitalRead(przod)==LOW)   {
81   Komunikacja(7,1);
82   Komunikacja(8,1);
83     while(digitalRead(przod)==LOW)
84     {          } }
```

Fig. 4. Code for the master microcontroller – forward drive

```
87    if(digitalRead(tyl)==LOW){
88   Komunikacja(7,2);
89   Komunikacja(8,2);
90      while(digitalRead(tyl)==LOW)
91     {          }}
```

Fig. 5. Code for the master microcontroller – reverse drive

```
102 if(digitalRead(hamulec)==LOW)
103   {
104      KomunikacjaZnak(7,"h ");
105      KomunikacjaZnak(8,"h ");
106
107     while(digitalRead(hamulec)==LOW)
108     {          }
109      KomunikacjaZnak(7,"q ");
110      KomunikacjaZnak(8,"q ");
111   }
```

Fig. 6. Code for the master microcontroller – braking

The code for reverse drive and braking differs only in the data transmitted by the bus and the functions that are called by different buttons on the user console.

The microcontroller controls motor speed through the potentiometer on the user console. The code checks the value at the analog input connected to the potentiometer (Fig. 7).

```
65  predkoscAktualna = analogRead(potencjometr);
66  predkoscAktualna = map(predkoscAktualna, 0, 1023, 3, 255);
67
68  if(predkoscAktualna != predkoscPoprzednia)
69  {
70    predkoscPoprzednia = predkoscAktualna;
71
72      Komunikacja(7,predkoscAktualna);
73      Komunikacja(8,predkoscAktualna);
74  }
```

Fig. 7. Code for the master microcontroller – speed control

Values in the range of 0-1023 are stored in the *currentSpeed* variable. In the next step, the values are converted by the map function to the range of 3-255. The *if* statement checks potentiometer value. If potentiometer value has changed, the respective function is called to transmit information about motor speed to slave microcontrollers.

If the user wants to turn the vehicle and presses the left turn button, the motor speed value that is transmitted to the right motor controller is halved. As a result, the left motor turns at half the speed of the right motor, which causes the vehicle to turn in a given direction. The left turn code is presented in Figure 8, and the right turn code is presented in Figure 9.

```
125   if(digitalRead(lewo)==LOW)
126   {
127    Komunikacja(7,predkoscAktualna/2);
128     while(digitalRead(lewo)==LOW)
```

Fig. 8. Code for the master microcontroller – left turn

```
116 if(digitalRead(prawo)==LOW)
117  {
118    Komunikacja(8,predkoscAktualna/2);
119       while(digitalRead(prawo)==LOW)
```

Fig. 9. Code for the master microcontroller – right turn

The information sent by the master microcontroller to slave microcontrollers is composed of values in the range of 0 to 255 and ASCII characters. Each value and character carries specific information:

1 – forward drive,
2 – reverse drive,
3-255 – motor speed,
'h' – engage brake,
'q' – release brake,
'u' – unipolar motor control,
'b' – bipolar motor control.

Code for slave microcontrollers

The code for slave microcontrollers that drive motors contains functions responsible for:

– switching transistors in a desired commutation sequence at the corresponding pulse width modulation (PWM) values,

– defining rotor position,

– receiving information about operating parameters from the master microcontroller.

Electronic commutation

The main purpose of a slave microcontroller is to replace a mechanical commutator with an electronic commutator. Transistors are switched in a given commutation sequence to pass current through motor windings and to set the rotor in motion. The commutation sequences for transistor switches are presented in Table 1.

Table 1

Commutation sequences for transistor switches when the motor turns right (clockwise)

	1	2	3	4	5	6
AH	1	0	0	0	0	1
BH	0	1	1	0	0	0
CH	0	0	0	1	1	0
AL	0	0	1	1	0	0
BL	0	0	0	0	1	1
CL	1	1	0	0	0	0

Microprocessor outputs AH, BH and CH supply upper transistor switches. Microprocessor outputs AL, BL and CL supply lower transistor switches. Numbers 1 to 6 in the top row of the table represent commutation steps. Two transistors are excited in each step. The commutation sequence has to be reversed to turn the motor in the left (counterclockwise) direction. The code for the function that switches transistors is presented in Figure 10.

```
16 void NastepnyKrokKomutacji(int krok){
17       switch(krok){
18              case 0:   AH_BL();  break;
19              case 1:   AH_CL();  break;
20              case 2:   BH_CL();  break;
21              case 3:   BH_AL();  break;
22              case 4:   CH_AL();  break;
23              case 5:   CH_BL();  break;
24                }
25 }
```

Fig. 10. Code for slave microcontrollers – function for switching transistors

The function initiates one of the six commutation steps to pass current through windings. For example, step 0 calls function AH_BL () which powers the upper transistor A and the lower transistor B.

The function that powers transistors sets the value of microcontroller outputs to 9 and 6 to send a PWM signal to the transistor driver (Fig. 11). Pulses are generated by DC to AC inverters and by frequency converters. Pulse width is modulated by switching transistors on and off. This approach supports easy and high-precision control.

```
84 void AH_BL(){
85   analogWrite(9,Pwm1);
86   analogWrite(6,Pwm2);
87   analogWrite(10,0);
88   analogWrite(11,0);
89   analogWrite(3,0);
90   analogWrite(5,0);
91 }
```

Fig. 11. Code for slave microcontrollers – function that sets PWM value at microcontroller outputs

Determining rotor position in a BLDC motor

Information about rotor position is required for the correct operation of a brushless motor. The controller is equipped with a rotor position sensor. Hall effect sensors mounted on coil windings send information about rotor position to the microcontroller by outputting a value of 0 or 1.

The combination of values output by three sensors describes rotor position (Fig. 12). When rotor position has been detected and the time of the next commutation step has been determined, the program calls a function that activates the next transistor pair.

```
194          stanHallA = digitalRead(hallA);
195          stanHallB = digitalRead(hallB);
196          stanHallC = digitalRead(hallC);
```

Fig. 12. Code for slave microcontrollers – values output by Hall effect sensors

The code that checks the combination of values output by Hall effect sensors is presented in Figure 13. If sensors C and A output 1, the *outputHall* variable is assigned a value that corresponds to the next commutation step.

```
42 void wykrywaniePolozeniaPrawo() {
43   if ((stanHallC == 1) && (stanHallB == 0) && (stanHallA == 1)) {
44     stan = 0;
45   }
46   if ((stanHallC == 0) && (stanHallB == 0) && (stanHallA == 1)) {
47     stan = 1;
48   }
49   if ((stanHallC == 0) && (stanHallB == 1) && (stanHallA == 1)) {
50     stan = 2;
51   }
52   if ((stanHallC == 0) && (stanHallB == 1) && (stanHallA == 0)) {
53     stan = 3;
54   }
55   if ((stanHallC == 1) && (stanHallB == 1) && (stanHallA == 0)) {
56     stan = 4;
57   }
58   if ((stanHallC == 1) && (stanHallB == 0) && (stanHallA == 0)) {
59     stan = 5;
60 }}
```

Fig. 13. Code for slave microcontrollers – detection of rotor position

I²C communication and motor control

The information about motor speed and direction is transmitted by the master microcontroller via the I²C data bus. The relevant data is sent in the form of values and ASCII characters. The slave microcontroller receives data and calls the corresponding functions. The code that enables slave microcontrollers to receive data is presented in Figure 14.

```
27 void odbierzInformacje(int howMany){
28   while (1 < Wire.available()) {
29   c = Wire.read();
30   if(c =='u'){metoda=1; } // metoda true unipolarna
31   if(c =='b'){metoda=2;  } // metoda false bipolarna
32   if(c =='h'){hamulecWlacz();}
33   if(c== 'q'){hamulecWylacz(); c='r';}
34   }
35   x = Wire.read();
36     if(x==1){kierunekObrotow=2;}    //lewo
37     if(x==2){kierunekObrotow=1; }   // prawo
38     if(x>=3 ){predkosc=x; predkosc=map(predkosc,3,255,10000,1000);}
39 }
```

Fig. 14. Code for slave microcontrollers – receiving data transmitted by the I²C bus

The *receiveInformation* function is called when information is received. The transmitted information is composed of ASCII characters such as *u*, *b, h* and *q*, and values in the range of 0 to 255. The *if* statement in the function code defines the direction of motor rotation. The unipolar or bipolar control mode has to be selected when the controller is activated. Character *u* denotes the unipolar mode, and character *b* denotes the bipolar mode. The direction of motor rotation which determines the direction of drive is determined in the next step. Value 1 indicates that the motor is moving in the right (clockwise) direction, and value 2 indicates that the motor is moving in the left (counterclockwise) direction. Values higher or equal to 3 denote motor speed. The higher the value, the faster the commutation sequence. Character *h* denotes electric braking, and the microcontroller activates two motor windings until character *q* is received.

Summary

The designed controller for brushless DC motors has been patented (SYROKA, KRAJEWSKI 2019) for use in commercial applications. The device can be applied in electric vehicles for controlling BLDC motors mounted in wheels. It was

developed as part of a research project at the University of Warmia and Mazury in Olsztyn (Books – Digital Control, SYROKA 2019) dedicated to the construction of electric vehicles and electric drives that rely on renewable sources of energy.

References

ÅSTROM K.J., MURRAY R.M. 2006. *An Introduction for Scientists and Engineers*. Feedback Systems. Princeton University Press, Princeton, Oxford.

BOLTON W. 2006. *Programmable Logic Controllers*. Elsevier, Amsterdam.

BUSO S., MATTAVELLI P. 2006. *Digital Control in Power Electronics*. Morgan & Claypool, Publisher, San Rafael, CA.

CHEN C.-T. 1991. *Analog and Digital Control system Design: Transfer Function, State Space, and Algebraic Methods*. Saunders College Publishing, Filadelfia, Pensylwania.

DENTON T. 2016. *Electric and Hybrid Vehicles*. Routledge, London.

DORF R.C., BISHOP R.H. 2008. *Modern Control System Solution Manual*. Prentice Hall, Upper Saddle River, New Jersey.

EMADI A., ALIREZA K., ZHONG N., YOUNG J.L. 2009. *Integrated Power Electronic Converters and Digital Control*. CRC Press LLC, Boca Raton, Florida.

EMADI A., EHSANI M., MILLER J.M. 2004. *Power Systems, Land, Sea, Air, and Space Vehicles*. Marcel Dekker, New York.

FADALI S. 2009. *Digital Control Engineering, Analysis and Design*. Elsevier, Amsterdam.

FEUER A., GOODWIN G.C. 1996. *Sampling in Digital Signal Processing and Control*. Brikhauser, Bazylea.

GABOR R., KOWOL M., KOŁODZIEJ J., KMIECIK S., MYNAREK P. 2019. *Switchable reluctance motor, especially for the bicycle*. Patent No 231882.

GLINKA T., FRĘCHOWICZ A. 2007. *Brushless DC motor speed control system*. Patent No.P195447.

IQBAL H. 2003. *Electric and Hybrid Vehicles, Design Fundamentals*. CRC Press LLC, Boca Raton, Florida.

JONGSEONG J., WONTAE J. 2019. *Method of controlling constant current of brushless dc motor and controller of brushless dc motor using the same*. United States Patent Application Publication, US2018323736 (A1).

KHAJEPOUR A., FALLAH S., GOODARZI A. 2014. *Electric and Hybrid Vehicles Technologies, Modeling and Control: a Mechatronic Approach*. John Wiley & Sons Ltd, Hoboken, New Jersey.

KOJIMA, N., ANNAKA T. 2019. *Motor control apparatus and motor unit*. United States Patent Application Publication, US2019047517 (A1).

KOLANO K. 2020. *Method for measuring the angular position of the shaft of a brushless DC motor with shaft position sensors*. Patent No.P235653.

LANDAU I.D., ZITO G. 2006. *Digital Control Systems Design, Identification and Implementation*. Springer, Berlin.

LUECKE J. 2005. *Analog and Digital Circuits for Electronic Control System Applications Using the TI MSP430 Microcontroller*. Elsvier, Amsterdam.

MI CH., MASRUR M.A., GAO D.W. 2011. *Hybrid Electric Vehicles Principles and Applications with Practical Perspectives*. John Wiley & Sons Ltd, Hoboken, New Jersey.

MOUDGALYA K.M. 2007. *Digital Control*. John Wiley & Sons Ltd., Hoboken.

MURRAY R.M., LI Z., SHANKAR SASTRY S. 1994. *A Mathematical Introduction to Robotic Manipulation*. CRC Press LLC, Boca Raton, Florida.

OGATA K. 1995. *Discrete Time Control Systems*. Prentice-Hall, Upper Saddle River, New Jersey.

PISTOIA G. 2010. *Electric and Hybrid Vehicles Power Sources, Models, Sustainability, Infrastructure and the Market*. Elsevier, Amsterdam.

SIKORA A., ZIELONKA A. 2011. *Power supply system for a BLDC motor*. Patent No. P.394971.
ŚLUSAREK B., PRZYBYLSKI M., GAWRYŚ P. 2014. *Hall effect sensor of the shaft position of the brushless DC motor*. Patent No.P218476.
SOYLU S. 2011. *Electric Vehicles – the Benefits and Barriers*. IntechOpen, London.
STARR G.P. 2006. *Introduction to Applied Digital Control*. Springer, Berlin.
STEVIĆ Z. 2012. *New Generation of Electric Yehicles*. IntechOpen, London.
SYROKA Z., KRAJEWSKI K. 2019. *Controller for brushless DC motors*. Patent No. P431380. Filing date: 4 October 2019.
SYROKA Z.W. 2019. *Electric Vehicels – Digital Control*. Scholars' Press, Riga.
WILLIAMSON S.S. 2013. *Energy Management Strategies for Electric and Plug-in Hybrid Electric Vehicles*. Springer, Berlin.