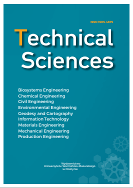
Technical Sciences, 2023, 26, 17–31

Yearbook peer-reviewed scientific journal

ISSN 1505-4675
e-ISSN 2083-4527

TECHNICAL SCIENCES

Homepage: https://czasopisma.uwm.edu.pl/index.php/ts/



DOI: https://doi.org/10.31648/ts.8258

AN OPTOELECTRONIC SYSTEM FOR CONTROLLING A DIRECT CURRENT MOTOR. PART 2: SOFTWARE

Zenon Syroka

ORCID: 0000-0003-3318-8495
Faculty of Technical Sciences
University of Warmia and Mazury in Olsztyn

Received 2 October 2022; accepted 18 December 2022; available online 19 December 2022.

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

A b s t r a c t

The software for an optoelectronic system for controlling a direct current (DC) motor is presented in Part 2 of the article. The structure of the designed control system was described in Part 1. The developed system processes data received from an infrared transmitter. The project was upgraded in successive stages of development, and it ultimately evolved into a small computer with a motor controller. The designed system automatically adjusts the motor's rotation and speed. The user is tasked only with conveying operational commands. The entire system is based on a single microcontroller.

The designed optoelectronic system receives user commands (the program can be modified to support free-space optical communication networks conforming to all communication standards). The system activates the motor, counts the number of rotations and adjusts the motor's position.

The designed system operates on the following principle: the user sends commands to the motor via a remote control with an infrared diode. The keys on the remote control have been programmed with different commands. The transmitted data are processed by the system which activates the motor and sets the desired motor speed. The task is completed, and the system is ready to process the next command. If the number of rotations differs from the preset value, the motor's position is adjusted. If the physical position of the rotor axis is altered, the system corrects the offset to the last programmed position. The designed system can be easily adapted to various types of motors and IR controllers.

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

Introduction

Wireless infrared devices for controlling direct current (DC) motors use free-space optical communication networks. Free-space optical data links rely on electromagnetic radiation in the visible or the infrared spectrum which is sent by the transmitter to the receiver, where electric charge – the control signal – is generated by a stream of photons. The signal is then amplified, demodulated and processed. In this solution, the signal is carried by the free space between the transmitter and the receiver. Various sources of electromagnetic interference with a wavelength similar to the wavelength at which the transmitter and the receiver operate can limit the applicability of the above solution. In indoor premises, electromagnetic interference is generated by heat sources, lamps and sunlight, and the number of potential disruptors is likely to increase over longer communication distances. Despite the above, numerous devices can be implemented in free-space optical communication systems, including distance sensors, slotted optical switches with aperture width of several millimeters to more than 10 centimeters, remote controls for household equipment with a range of several meters, data transmission systems used by telecommunications companies, such as VLC and FSO with data transmission speeds of up to 10 GB/s and a range of more than 10 kilometers, as well Lunar Laser Communication Demonstration (LLCD) which relies on laser light to communicate with a receiver station across a distance of several hundred kilometers (385 km).

The aim of this study was to develop and optoelectronic system for controlling a DC motor. A system that processes data from an infrared transmitter and handles tasks broken into data frames was built. The project was upgraded in successive stages, and it ultimately evolved into a small computer with a motor controller. The designed system automatically adjusts the motor's revolution and speed. The user is tasked only with conveying operational commands.

The device receives user commands (the program can be modified to support free-space optical communication networks conforming to all communication standards). The system activates the motor, counts the number of rotations and adjusts the motor's position. The system features optoelectronic components, a DC motor and devices for controlling the motor.

The user sends commands to the motor via a remote control with an infrared diode. The keys on the remote control have been programmed with different commands. The transmitted data are processed by the system which activates the motor and sets the desired motor speed. The task is completed, and the system is ready to process the next command. If the number of revolutions differs from the preset value, the motor's position is adjusted. If the physical position of the rotor axis is altered, the system corrects the offset to the last programmed position. The designed system can be easily adapted to various types of motors and IR controllers.

A patent application for the developed system has been submitted to the patent office.

Development of the programming algorithm

The designed system for controlling a DC motor with the use of an IR remote control should be reliable and easy to modify. The keys on the remote control are programmed with different functions, and the user is tasked only with pressing the respective key to execute a given command.

The IRremote.h library was used to write the code.

The developed software had to be easy to modify, and two programs were written for this purpose. The first program reads the value of the signal sent by the remote control, and the second program reads the duty cycle of the PWM signal.

In the first step, the IR receiver is configured and the IRremote.h library is implemented. The following commands are used to implement the library:

– IRrecv irrecv(irDiode) – create the receiver object and assign a name to the receiver (irDiode);

– decode_results result – decode the received data. The program returns “true” when a code is received, and “false” when a code is not received. When a code is received, data are stored in “results”;

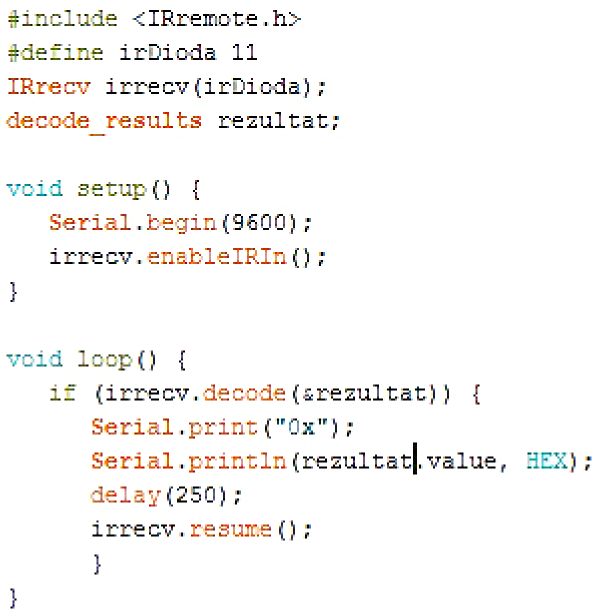
– irrecv.enableIRIn() – begin the receiving process. This command will enable the timer interrupt which consumes a small amount of CPU every 50 microseconds;

– irrecv.resume() – when a code is received, this command must be called to reset the receiver and prepare it to receive another code.

The code for reading the remote control signal is presented in Figure 1. When data have been read, their value is displayed in hexadecimal base with the 0x prefix. This is followed by a 0.25 s interval during which the receiver is reset and prepared to receive another code.

Digital output pins in a microcontroller have fixed values; therefore, a motor’s rotational speed cannot be controlled by changing voltage. This is why a PWM signal was used. The duty cycle of a PWM signal can vary over time. The force of inertia acting on a rotating stator does not stop the motor immediately; therefore, the motor’s rotational speed can be controlled. However, the motor will not be activated when the speed is too low. The value of a PWM signal can be higher than rated voltage. Average voltage cannot exceed rated voltage.

The minimum duty cycle of a PWM signal can vary in motors with the same parameters, in particular in geared motors. The system was modified by developing a program that displays the minimum duty cycle on an integrated serial monitor.



```
#include <IRremote.h>
#define irDioda 11
IRrecv irrecv(irDioda);
decode_results rezultat;

void setup() {
   Serial.begin(9600);
   irrecv.enableIRIn();
}

void loop() {
   if (irrecv.decode(&rezultat)) {
      Serial.print("0x");
      Serial.println(rezultat.value, HEX);
      delay(250);
      irrecv.resume();
      }
}
```

Fig. 1. Code for reading the signal from the remote control

A potentiometer was connected to the microprocessor's analog-to-digital converter. The microprocessor's resistance was changed to modify the signal's duty cycle and determine the minimum duty cycle that will activate the motor.

The code for determining the minimum duty cycle of a PWM signal is presented in Figure 2.

The direction of each pin has to be determined in every program. The following commands were used:

– byte readValue = 0 – determine the value read from the analog-to-digital converter;

– Serial.begin(9600) – establish communication to display the value on the serial monitor by setting the UART baud rate;

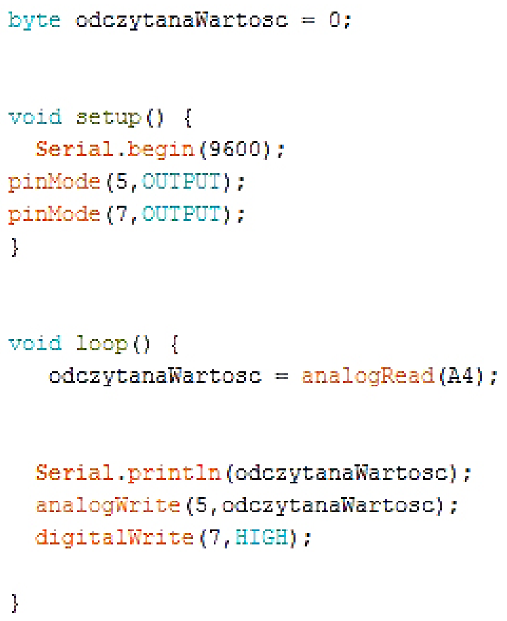
– readValue = analogRead(A4) – read the value from the analog-to-digital converter;

– Serial.println(readValue) – transmit the measured voltage;

– analogWrite(5,readValue) – the read value is the duty cycle.

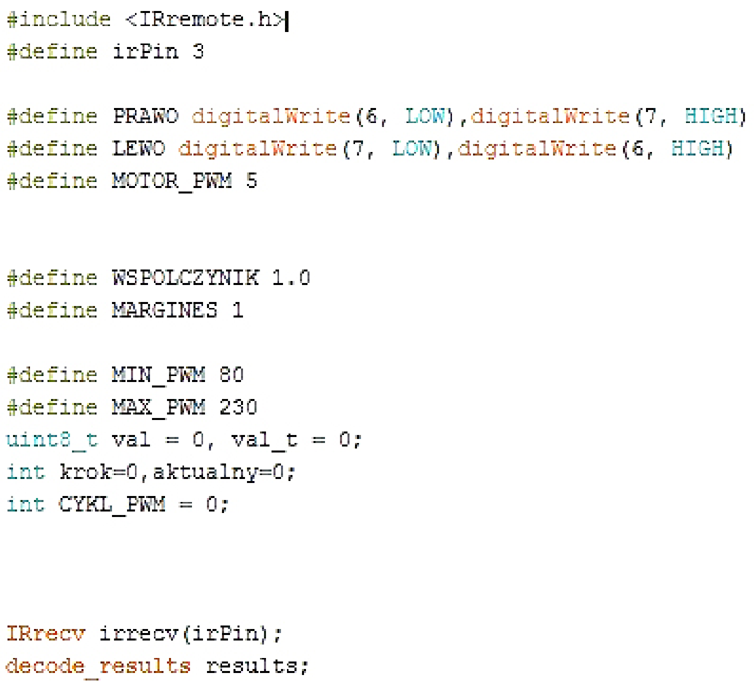
The main program receives the data sent by the remote control, processes that data and executes the relevant command. The program also monitors the steps performed by the motor and displays the relevant information on the computer screen.

The lines of the code that define variables and parse data are presented in Figure 3.



```
byte odczytanaWartosc = 0;


void setup() {
  Serial.begin(9600);
pinMode(5,OUTPUT);
pinMode(7,OUTPUT);
}


void loop() {
   odczytanaWartosc = analogRead(A4);


  Serial.println(odczytanaWartosc);
  analogWrite(5,odczytanaWartosc);
  digitalWrite(7,HIGH);

}
```

Fig. 2. Code for determining the minimum duty cycle of a PWM signal



```
#include <IRremote.h>
#define irPin 3

#define PRAWO digitalWrite(6, LOW),digitalWrite(7, HIGH)
#define LEWO digitalWrite(7, LOW),digitalWrite(6, HIGH)
#define MOTOR_PWM 5


#define WSPOLCZYNIK 1.0
#define MARGINES 1

#define MIN_PWM 80
#define MAX_PWM 230
uint8_t val = 0, val_t = 0;
int krok=0,aktualny=0;
int CYKL_PWM = 0;



IRrecv irrecv(irPin);
decode_results results;
```

Fig. 3. Code lines which define variables and parse data

Peripheral devices are configured and variables are defined in the first step of the program. Selected pins were defined and configured to default settings to increase the program's readability. The following configuration settings were used:

– IRremote.h – library used to configure the IR receiver;

– irPin 3 – pin 3 was defined as irPin which receives data;

– RIGHT/LEFT – connections responsible for the direction of motor rotation;

– MOTOR_PWM 5 – pin 5 was defined as MOTOR_PWM which generates a PWM signal;

– GAIN 1.0 – proportional gain of the control loop mechanism;

– MARGIN – upper limit of the proportional controller denoting the maximum difference in motor's rotational speed;

– MIN_PWM/MAX_PWM – minimum and maximum values of PWM set by the program;

– val/val_t – variables denoting the previous value and the current value, which are compared in successive steps of the program;

– step – variable denoting the number of steps to be executed. When data are received and decoded, the step is set to the value that is assigned to a given data packet;

– current – variable denoting the current number of executed steps. This value is compared with the preset value;

– CYCLE_PWM – variable denoting the value of PWM which is used to stop the motor;

– IRrecv irrecv(irPin) – create the receiver object and assigned a name to the receiver (irPin);

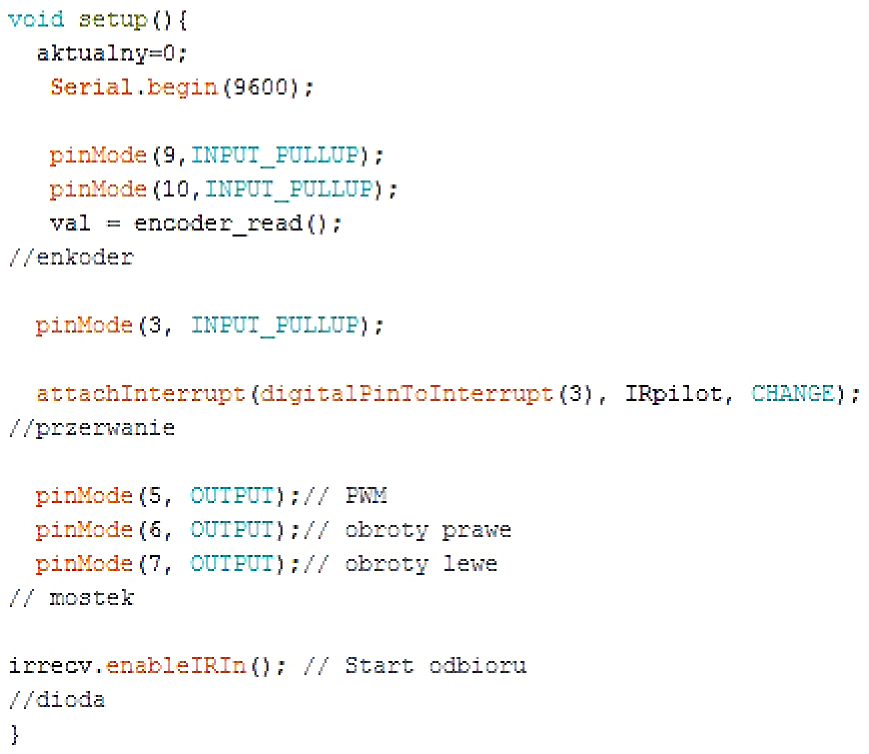
– decode_results results – decode data. The program returns "true" when a code is received, and "false" when a code is not received. When a code is received, data are stored in "results".

The setup() function is called automatically when the microcontroller is activated or the reset key is pressed. This function is used to set system values, such as initial variable values, and to declare inputs and outputs. The setup() function does not return values and parameters are not input in the function.

In the developed program, the setup() function was used to set the initial values of variables, declare microcontroller inputs and outputs, establish communication, display values on the serial monitor by setting the UART baud rate, and receive information from the remote control. The function defines the input responsible for interrupts, interrupt subroutines and the effects of the interrupt service routine.

An interrupt is a signal from a peripheral device that interrupts the main program and jumps to a specific function entry inside the interrupt service routine.

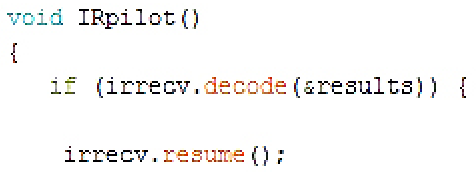
A fragment of the main program with the setup() loop is presented in Figure 4.



```
void setup(){
  aktualny=0;
   Serial.begin(9600);

   pinMode(9,INPUT_PULLUP);
   pinMode(10,INPUT_PULLUP);
   val = encoder_read();
//enkoder

  pinMode(3, INPUT_PULLUP);

  attachInterrupt(digitalPinToInterrupt(3), IRpilot, CHANGE);
//przerwanie

  pinMode(5, OUTPUT);// PWM
  pinMode(6, OUTPUT);// obroty prawe
  pinMode(7, OUTPUT);// obroty lewe
// mostek

irrecv.enableIRIn(); // Start odbioru
//dioda
}
```

Fig. 4. A fragment of the main program with the setup() loop

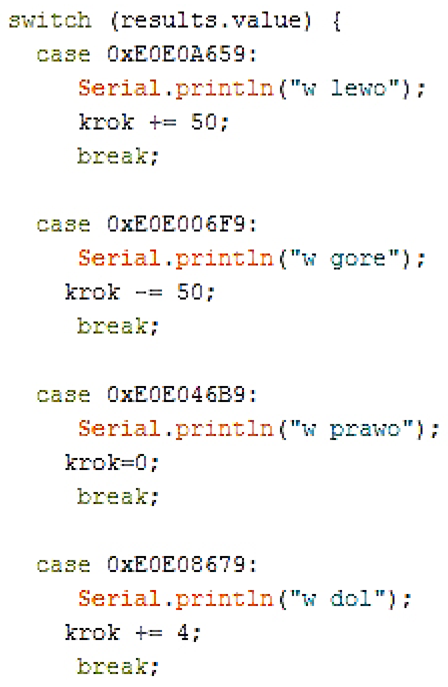
When the signal from pin 3 changes, the main program is interrupted and the stored instructions are executed. The received data are decoded. The beginning of the interrupt subroutine is initiated is presented in Figure 5.



```
void IRpilot()
{
   if (irrecv.decode(&results)) {

    irrecv.resume();
```

Fig. 5. Beginning of the interrupt subroutine

The decoded data frame is compared with the data in the switch case conditional statement. When the received signal matches the declared variable value, the value of the step variable is changed, and subsequent instructions are executed in the main program. If the received signal differs from the declared variable, the program waits for the next value. This solution eliminates the influence of the external environment on the received data and enables the user to program other keys on the remote control. Devices manufactured by different companies and conforming to various standards can be used by changing the switch case values. Signal values are read by the described program. The switch case conditional statement is presented in Figure 6.



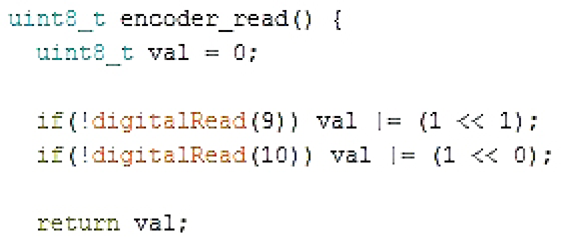
```
switch (results.value) {
  case 0xE0E0A659:
     Serial.println("w lewo");
     krok += 50;
     break;

  case 0xE0E006F9:
     Serial.println("w gore");
    krok -= 50;
     break;

  case 0xE0E046B9:
     Serial.println("w prawo");
    krok=0;
     break;

  case 0xE0E08679:
     Serial.println("w dol");
    krok += 4;
     break;
```

Fig. 6. Switch case conditional statement

The main loop contains functions that execute the preset number of steps. Conditional statements that monitor the number of motor rotations were used to perform this task.

An algorithm that counts the number of steps from the optical encoder was developed with the use of Gray code. Gray code is a binary numeral system where two successive values differ in only one bit. It is applied when logic states change, which does not occur simultaneously.

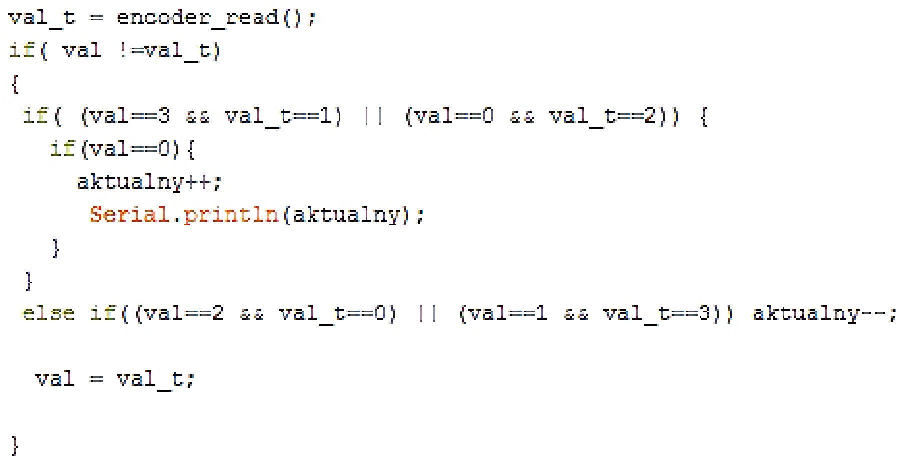
The position of digits is set by the encoder which detected the change. The logic states of optocouplers generate a binary number which assumes the value of 0 to 3 in the decimal system. The same values appear cyclically, and their sequence is determined by the direction of the motor's rotation. These values are used to determine the direction in which the motor rotates and the number of rotations. Gray code in the main program is presented in Figure 7.



```
uint8_t encoder_read() {
  uint8_t val = 0;

  if(!digitalRead(9)) val |= (1 << 1);
  if(!digitalRead(10)) val |= (1 << 0);

  return val;
```

Fig. 7. Gray code in the main program

The state of pins 9 and 10 is examined. Val is an 8-bit variable, but only 2 bits are used. If pin 9 has a falling edge, the first bit is set at 1. If pin 10 has a falling edge, the zero bit is set at 1. The value is returned by the “return” command. The value of variable val in the decimal system is presented in Figure 8.

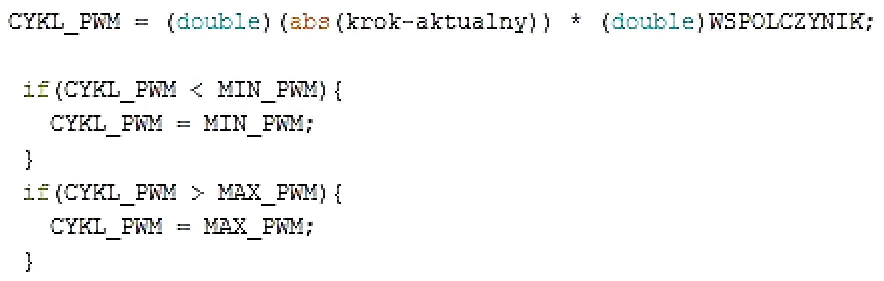


```
val_t = encoder_read();
if( val !=val_t)
{
 if( (val==3 && val_t==1) || (val==0 && val_t==2)) {
   if(val==0){
     aktualny++;
      Serial.println(aktualny);
   }
 }
 else if((val==2 && val_t==0) || (val==1 && val_t==3)) aktualny--;

  val = val_t;

}
```

Fig. 8. Algorithm counting the number of pulses

The current value of the encoder is read in the next step. If the current value differs from the previous value, the system checks whether the previous value was 3 and the current value is 1, or whether the previous value was 0 and the current value is 2. If the previous value was 0, the value of the current step is increased and displayed on the serial monitor. The value of the current step is decreased if the previous value was not 0. After this step, the previous value equals the current value.

In the following step, the value of the CYCLE_PWM variable is calculated with the use of the code presented in Figure 9. The absolute difference between the preset steps and the current steps is multiplied by the proportional gain of the control loop mechanism. If CYCLE_PWM is lower than the minimum value of PWM, it assumes the value of MIN_PWM. If CYCLE_PWM is higher than the maximum value of PWM, it assumes the value of MAX_PWM.



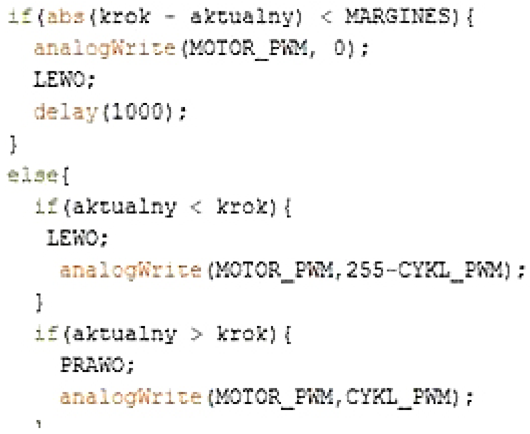
```
CYKL_PWM = (double)(abs(krok-aktualny)) * (double)WSPOLCZYNIK;

  if(CYKL_PWM < MIN_PWM){
    CYKL_PWM = MIN_PWM;
  }
  if(CYKL_PWM > MAX_PWM){
    CYKL_PWM = MAX_PWM;
  }
```

Fig. 9. Code for calculating the value of the CYCLE_PWM variable

The program then calculates the absolute difference between the preset steps. If the difference is smaller than the MARGIN variable, PWM equals zero, the motor is stopped and the program pauses for 1 s. If the result is different, two scenarios are possible (Fig. 10):

– if the number of current steps is smaller than the number of preset steps, the motor rotates counterclockwise, the value of PWM is the difference between 255 (the maximum value) and CYCLE_PWM. The motor's speed decreases before a full stop;

– the motor rotates clockwise.



```
if(abs(krok - aktualny) < MARGINES){
  analogWrite(MOTOR_PWM, 0);
  LEWO;
  delay(1000);
}
else{
  if(aktualny < krok){
   LEWO;
    analogWrite(MOTOR_PWM,255-CYKL_PWM);
  }
  if(aktualny > krok){
    PRAWO;
    analogWrite(MOTOR_PWM,CYKL_PWM);
  }
```

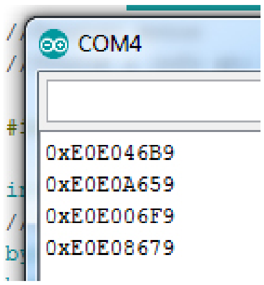
Fig. 10. Code responsible for motor rotation

Device configuration and tests

The device has to be configured before tests. Remote control keys are programmed, and the value of MIN_PWM which activates the rotor shaft is set. These tasks were accomplished with the use of:

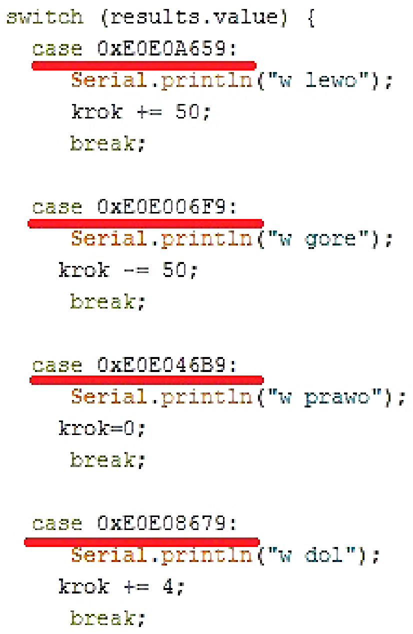
– a program that reads the signal from the remote control;
– a program that sets the minimum value of the duty cycle.

The device was connected to a computer and power supply, and the program reading the signal from the remote control was uploaded to the microcontroller. The serial monitor was turned on, and four selected keys were pressed. The displayed values (Fig. 11) were input in the appropriate lines of the code (Fig. 12).



```
0xE0E046B9
0xE0E0A659
0xE0E006F9
0xE0E08679
```

Fig. 11. Values displayed on the serial monitor

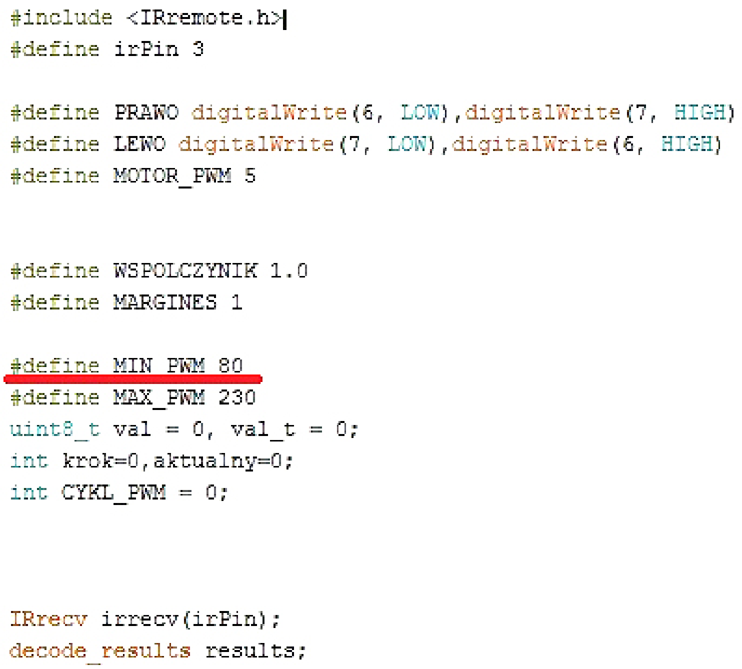


```
switch (results.value) {
  case 0xE0E0A659:
     Serial.println("w lewo");
     krok += 50;
     break;

  case 0xE0E006F9:
     Serial.println("w gore");
    krok -= 50;
     break;

  case 0xE0E046B9:
     Serial.println("w prawo");
    krok=0;
     break;

  case 0xE0E08679:
     Serial.println("w dol");
    krok += 4;
     break;
```

Fig. 12. Code lines for entering the displayed values

To determine the minimum duty cycle, the program was uploaded to the microcontroller, and the serial monitor was turned on. Before this step, a potentiometer was connected to the microcontroller's A4 output and the system was powered. The potentiometer should be set to a minimum value upon connection.

The potentiometer dial was slowly rotated when the motor began to move. The potentiometer was then restored to the lowest value, and the test was repeated several times. The average result was entered in the appropriate line of the code (Fig. 13).



```
#include <IRremote.h>
#define irPin 3

#define PRAWO digitalWrite(6, LOW),digitalWrite(7, HIGH)
#define LEWO digitalWrite(7, LOW),digitalWrite(6, HIGH)
#define MOTOR_PWM 5


#define WSPOLCZYNIK 1.0
#define MARGINES 1

#define MIN_PWM 80
#define MAX_PWM 230
uint8_t val = 0, val_t = 0;
int krok=0,aktualny=0;
int CYKL_PWM = 0;



IRrecv irrecv(irPin);
decode_results results;
```

Fig. 13. Code line for entering the minimum value of the PWM signal

The system was tested in the motor's operating environment. The motor's load was simulated with the worm gear. The tests were conducted to verify the initial assumptions and introduce the required modifications. The motor's operation was simulated in an indoor environment. The results were used to predict the motor's behavior during operation.

The system was supplied with power, connected to a computer, and a multimeter was connected to the IR receiver output. The program for reading data from the remote control was uploaded to the microcontroller. The impact of potentially disruptive factors that could cause errors in data acquisition was tested, including:

– fluorescent lamps;
– heaters;
– sunlight;
– 60 W incandescent light bulb;
– different remote control.

The monitored values were displayed on the serial monitor when the microcontroller did not communicate with the remote control. Minor voltage fluctuations were incidentally noted at the receiver output. Communication with a remote control operating in a different wireless standard was tested. The result was positive, which indicates that the designed system can be operated with the use of different remote controls or several remote controls.

The impact of the identified disruptions on the system's operation was tested in the next step. The main program was uploaded. After compilation, the program occupied 27% of the program memory and 22% of dynamic memory, which implies that the program can be expanded.

Data were transmitted correctly by the remote control, and the motor performed the desired number of rotations. The delay in data receipt did not exceed the delays noted in other devices that rely on the same technology. The only exception was the test where data transmission was disrupted by a different remote control. In this case, data receipt was delayed, but it had no effect on motor rotation. The above resulted from similar transmission wavelengths in IR remote controls.

The system's operation during command execution was tested. A command signal was sent, and a second signal was transmitted before the first command was executed. The value of the second signal was added to the value of the first signal. However, the current value and the time when the data frame was received remain unknown. A data receipt diode or a display presenting the number of motor rotations should be added in successive stages of system development. The previous value was subtracted during the transmission of data with a negative value. The motor rotated in the opposite direction if the transmitted value was higher than the previous value, which slightly affected the motor's stop position relative to the initial position. The maximum value approximated 20°, and it was determined by the encoder's structural design.

During operation, a motor can be subjected to vibration or other environmental factors that can affect the position of the rotor shaft. A minor change in the rotor angle was noted when manual force was applied. The motor began to adjust its rotor position when the displacement exceeded the boundary value of 20°, which is consistent with the initial assumptions. The rotor shaft also returned to its initial position when forcefully turned.

The number of motor rotations displayed on the serial monitor was checked during each test. Messages were displayed correctly. A display should be incorporated in successive stages of project development to monitor the system's performance without the use of a computer.

Conclusions

An optoelectronic system for controlling a DC motor was designed. The developed system processes data from the IR transmitter and handles tasks broken into data frames. The system receives user commands (the program can be modified to support free-space optical communication networks conforming to all communication standards), activates the motor, counts the number of rotations and adjusts the motor's position.

The designed system was built around an Arduino microcontroller which is relatively cheap, enables system modifications in the design stage and can be easily adapted during programming, which saves time and facilitates project performance. The Arduino environment can be also extended through the use of libraries which facilitate the incorporation of new modules, including modules without memory that can be easily adapted to the project's needs.

The developed system can be upgraded and modified in the future. The following solutions could be considered:

– the parameters of the DC motor could be displayed on an LCD screen;

– three motors could be mounted on the same frame to build a robotic arm or a plotter;

– a proportional integral derivative controller could be incorporated into the program to control motor rotation;

– the code for receiving user commands could be expanded;

– a rotating encoder with a larger number of slits could be applied to increase control precision;

– rotational speed could be measured during the motor's operation.

References

ALI E., KHALIGH A., NIE Z., LEE Y.J. 2009. *Integrated Power Electronic Converters and Digital Control*. CRC Press, Boca Raton.

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

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

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

DENTON T. 2016. *Electric and Hybrid Vehicles*. Routledge, San Diego.

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

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

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

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

GREGORY P. 2006. *Starr Introduction to Applied Digital Control*. Gregory P. Starr, New Mexico.

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

HUSAIN I. 2003. *Electric and Hybrid Vehicles, Design Fundamentals*. CRC Press LLC, Boca Raton, London.

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, Chichester.

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

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

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

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

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

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

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

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

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

SIKORA A., ZIELONKA A. 2011. *Power supply system for a BLDC motor*. Patent No. P.394971.

SOYLU S. 2011. *Electric Vehicles – the Benefits and Barriers*. Edited by Seref Soylu, Rijeka.

STEVIĆ Z. 2013. *New Generation of Electric Yehicles*. Edited by Zoran Stević, Rijeka.

SYROKA Z.W., JAKOCIUK D. 2020. *Battery recharging system in electric vehicle*. Patent No. P431380, filing date: 17 January 2020.

SYROKA Z.W. 2019. *Electric Vehicels – Digital Control*. Scholars' Press, Mauritius.

SYROKA Z.W., MERCHEL D. 2021. *Optoelectronic control system for an alternating current motor*. Patent decision of 5 February 2021; patent No. PL 236459.

ŚLUSAREK B. , PRZYBYLSKI M., GAWRYŚ P. 2014. *Hall effect sensor of the shaft position of the brushless DC motor*. Patent No.P218476.

WILLIAMSON S.S. 2013. *Energy Management Strategies for Electric and Plug-in Hybrid Electric Vehicles*. Springer, New York, London.