Software-hardware complex for controlling the power of CO2 laser radiation

D. D. Konnov

Student of the 6th year of the Faculty of Electronics, St. Petersburg State Electrotechnical University "LETI", the Russian Federation
With the assistance of V. F. Goncharov company Lumex http://www.lumex.ru/

Reviewer Ph.D. Associate Professor E. A. Smirnov
St. Petersburg State Electrotechnical University LETI, St. Petersburg, 197022 Russia
**e-mail: konnovd92@gmail.com*

Received April 30, 2017

Key words: laser cutting, CO_2-laser, dielectrics, dielectric cutting, CO_2-laser efficiency, Method of calibration of a gas-discharge tube, Linear Least Squares, Arduino, Delphi.

The article describes the private implementation of the software-hardware complex for adjusting the power of CO2 laser radiation using linear interpolation methods, as well as the method of calibration of a gas-discharge tube.

St. Petersburg, 2017

CONTENT

INTRODUCTION

Scientific and technological progress makes more stringent requirements for technological processing processes. In connection with this, it is justified to use computer and microprocessor technology to control the laser power in the operational mode, since it is a matter of thousandths of a second and a high accuracy of material processing.

At present, the sources of optical coherent, monochromatic radiation are an integral part of our life, and it is very important to regulate the power of the radiation incident on the workpiece during the processing of materials. There are basically two ways, the first of which involves adjusting the output radiation power through modulators, the second method is to change the output power by adjusting the laser active level. Both these methods can be automated, in the course of this work, an installation adapted to adjust the radiation power by limiting the pump current has been assembled and a software and hardware complex has been developed.

The source code of the application software is available at:
https://github.com/sharky92/LazerPowerControl

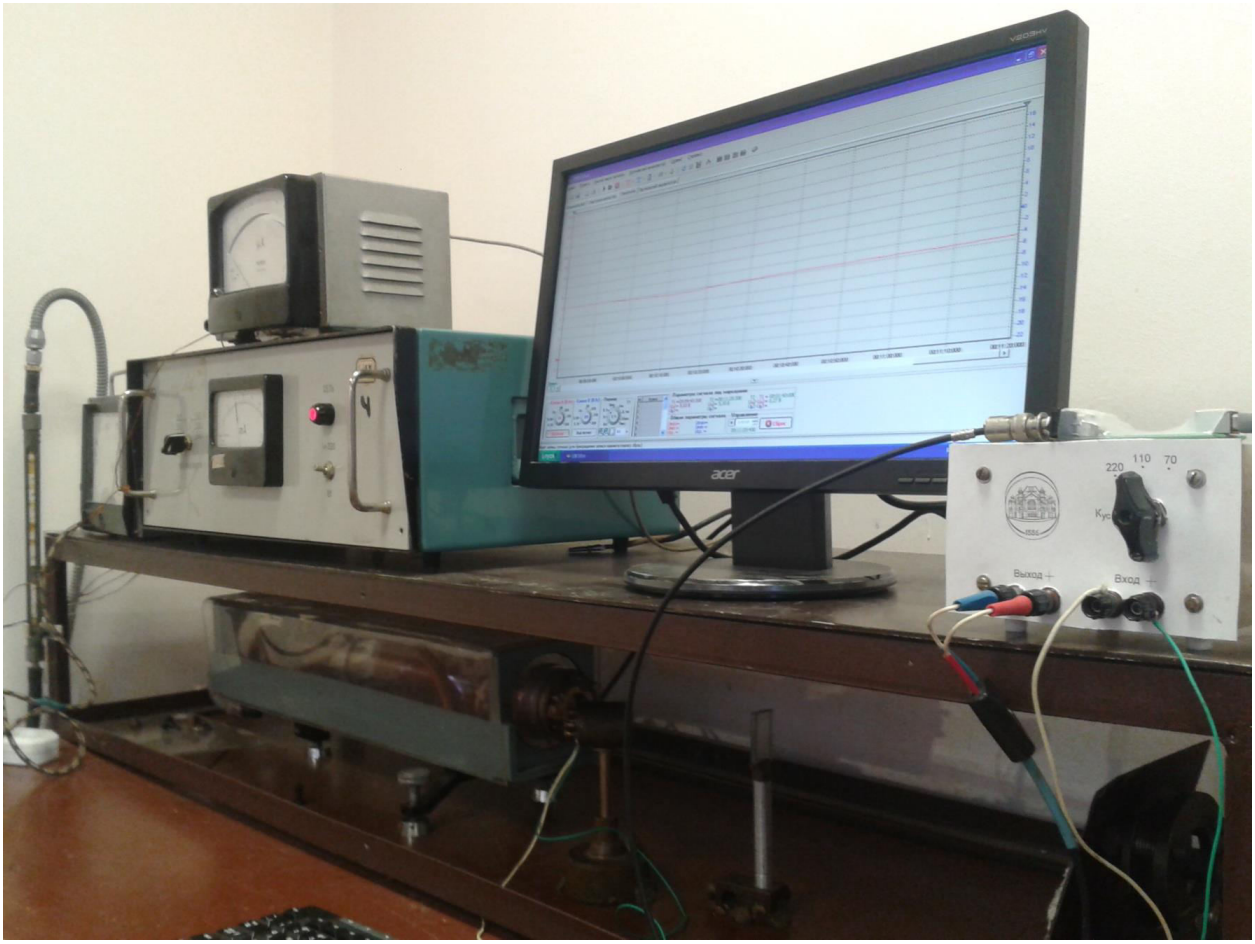

Fig. 1. Measuring unit

Fig. 2. Current modulation module

Fig. 3. Photoresistor

1. PURPOSE OF THE PROGRAM

The application program "Laser Power Control" is intended for automation of the laser processing of materials. The sequence of laser emission intensity levels for reproduction must first be recorded in a file of a certain format. In the technological cycle, the program reproduces the sequence of intensities specified in the file and controls the CO2 laser power in the online mode, by controlling the current in the gas-discharge tube circuit using a PWM (Pulse Width Modulation)-based DAC (ATmega328p).

The current control of the gas-discharge tube is controlled in accordance with the calibration curve in the computer's memory, which reflects the energy characteristic of this pre-calibrated tube. *see Graduation.*

The need for graduation of a gas discharge tube, substantiated in practice, is due to the floating characteristics of the tube itself, under the influence of such factors as: pump pressure, gradual depressurization, temperature factors during operation. In short, in practice, it has been established that a gas-discharge tube of a laser can not statically maintain its energy performance for a long time, so before operating a laser unit, it is recommended to calibrate its tube. *see Graduation.*

The program is developed with Object Pascal in Delphi 7 IDE. The algorithm is quite universal, and its application can be adapted to any process, from laser cutting and engraving to recording holograms.

The program performs the following main functions:

–Loading data from the file and processing the energy characteristic of the laser;

– Loading and processing the energy characteristic of control element;

– Loading and processing of operational data, i.e., a time scan for the radiation power to be obtained from the laser;

– PC COM port initialization;

– Start and stop the cutting process;

– Transfer of the control signal to the microcontroller in real time via a virtual USB COM port;

– Display of all parameters in real time.

1.1. Graduating (Methodology)

The energetic characteristic of a gas discharge tube is not a linear function and the exact shape of the curve depends on the characteristics of the particular tube.

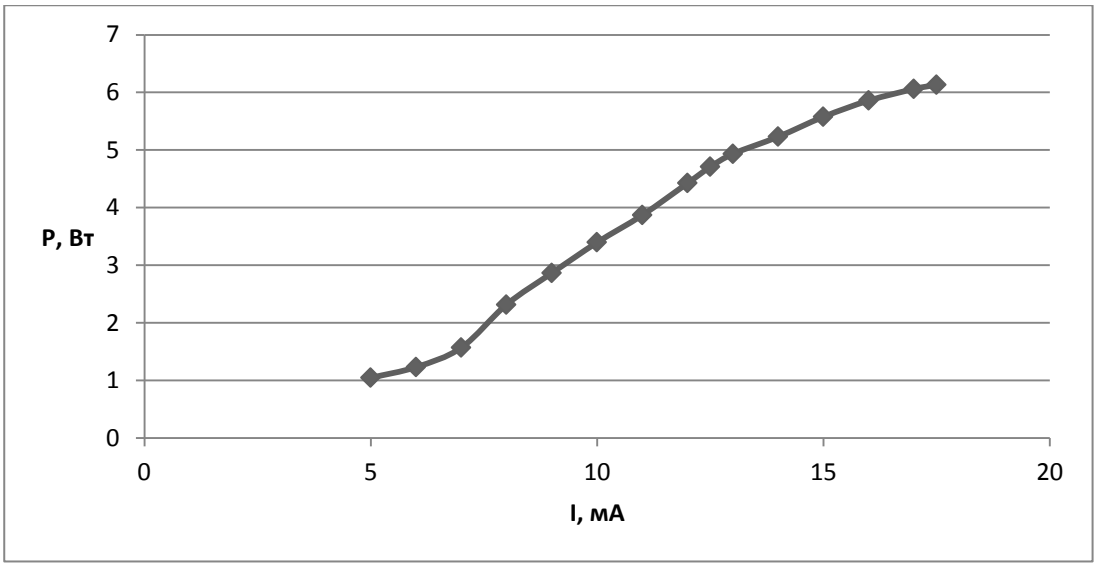

Fig. 4. Dependence of laser radiation power on discharge current

For proper operation of the hardware and software complex, it is necessary to provide experimental input data with which the correct operation in the online mode is performed. For this, it is necessary to calibrate the instrument. Graduating is done by laboratory measurement of the current of the gas-discharge tube and the extracted radiation power. To increase the accuracy of the calibration, multiple power measurements are necessary. In this case, it is required to perform this metering on the X axis at least 4 times with averaging the result along the Y axis. And when carrying out several measurements at one point, for example 4 times,

the accuracy of the final gradient curve will be higher. In this case, all 4 measurements can be considered equivalent and for each point calculate the average value of Y. According to statistics, the average value of N measurements is more accurate to the square root of N unit measurements. In our case N = 4 and the accuracy with respect to Y will be twice as high. In the process of searching for the value of X lying in the interval between the gradient points, in our case the necessary parameter must satisfy a simple mathematical relation: $(y - y_1)/(x - x_1) = (y_2 - y_1)/(x_2 - x_1)$, since the algorithm includes only two points. Hence from the given Y we find X: $x = [(x_2 - x_1)/(y_2 - y_1)*(y - y_1)] + x_1$. Approximation can be made for more points, left and right from the current on the graph, then the least squares method will be implemented, the program procedure LinearSquares(*see source code*) can take as an input parameter an array of more than 2 points.

2. OPERATION OF THE APPLICATION

2.1. Installation and startup

You do not need to install it to work with the program, just run the executable file. To load a subroutine into the microcontroller, you need to install on your PC free software provided by Arduino. It also allows you to interact with the microcontroller ATmega328p, which is part of the Arduino Software (IDE), which must be connected to the PC via a USB cable. The interaction of "Laser Power Control" application takes place in real time. The subroutine is loaded into the microcontroller using the standard Arduino IDE program. The Arduino subroutine itself is implemented in the C language for the Arduino family of controllers. The code of the subroutine with the description is presented further.

2.2. The interface of the program

Main window of "Laser Power Control" is shown in Fig. 5.

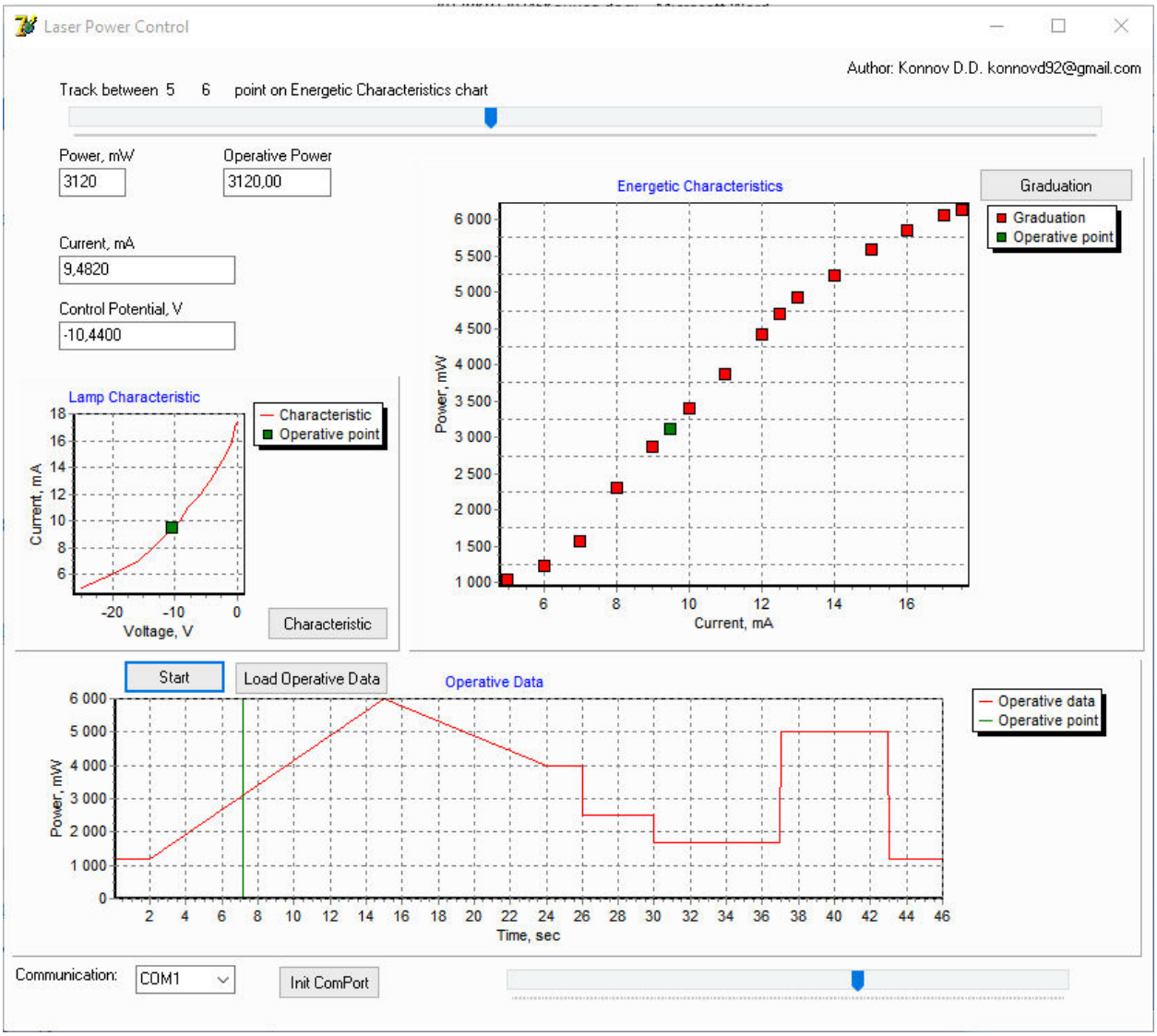

Fig. 5. Window of the "Laser Power Control"

Loading of all data into the program is performed by pressing the corresponding button and then selecting the required file in the appeared window. The file should be a text file consisting of an array of coordinates in the form of a column, where each line indicates the coordinates for one point so that first the value of X, then Y, separated by semicolon. The button "Graduation" loads the energy characteristic of the laser, the "Characteristic" button loads the characteristic of the control element, in our case it is the GU-50 generator lamp,

the "Load Operative Data" button loads the operational data file with the array of changes in the radiation power on the time scale. After that, by pressing the "Init Com Port" button, the COM port is initialized, you must first select the desired port, which can be found in Arduino IDE when connected to a PC via USB microcontroller. After pressing the "Start" button, the operation of the program is started, the necessary parameters are calculated and the event value is transferred to the control voltage for the microcontroller.

3. LOGIC OF THE PROGRAM

A complete listing of the program with detailed comments, through which you can understand the logic of the program, is at https://github.com/sharky92/LazerPowerControl.

Here, the key functional fragments of the program code are mentioned.

Function **LinearLeastSquares**

Implementation of the classical algorithm of the method of least squares. As an input parameter, an array of point values is accepted, and an array of two points can be minimally. To improve the approximation in the case of a linear function, the number of points can be arbitrarily more. If the function is nonlinear, then the nearest points will be chosen more precisely, as discussed above. The essence of the method is to find the minimum sum of squares of deviations for all points from the approximating straight line, thus the average statistical value is calculated. The output parameters are the coefficients ***m, b*** and ***r***, by means of which it is possible to describe a line that most satisfies the majority of points entering the array of the series, with ***r*** being the correlation coefficient; it characterizes how close the input array is to the linear dependence. With an input array of points that lie on one line, this parameter is equal to one. **LinearLeastSquares** is used in run-time mode to calculate the required value of X with the available input value Y of a point located between two known calibration points. In this case, the classical method of Linear Interpolation of Lagrange-Newton is realized.

Function **FindGradPoint**

The series usually consists of 20-40 reference points. This function returns the sequence number of the nearest reference point belonging to the series. The input value can be either X or Y, the additional parameter *pXorY* indicates what the input value is. This procedure serves as a preliminary tool, which allows us to find the equation of a straight line for the region where the operative point lies. After that, you can calculate the missing coordinate of the point, knowing the equation for it.

Function **LinearLeastSquaresForSerie**

Accepts as a parameter a series of TChart component from the graph, that is, the line from the graph and the point number. The output parameters are the coefficients ***m, b*** and ***r***. A series can consist of only twenty points, but after obtaining the coefficients ***m, b*** and ***r***, we can calculate the values of X and Y for any of its points. The reference (gradient) point, namely its number in the series, is searched using the function FindGradPoint, that is, the point that lies closest in the course of the motion of the function curve. Thus, we can always, with coefficients, calculate the value of Y for any input value X that belongs to this function.

Component TTimer

Movement on the timeline is implemented using the TTimer component. The operational data file with the array of changes in the laser emission intensity on the time scale is loaded into a series of graphs. File format, set of points: X - time stamp in seconds from the beginning of work, Y - value of power at this moment. The discretization of points is arbitrary, that is, there is no strict requirement for the value of Y every n sec / ms. The OnTimer event is called every 50 ms and the required power Y is automatically calculated on the available operational series. The current X value is stored in the vertical line Serie_OpChart_Vertical.XValue [0] and incremented by 0.06 every time the OnTimer event is called. That is, Serie_OpChart_Vertical.XValue[0] is used as a

global variable, we take Serie_OpChart_Vertical.XValue[0], incrementing it, Serie_OpChart_Vertical series is cleared

and call for Serie_OpChart_Vertical.AddXY(X, Y) remembers Serie_OpChart_Vertical.XValue[0] again.

After that using **LinearSquaresForSerie** for this X, the coefficients **M, B** and hence the value of Y in the operational series for the given X are calculated. When the maximum value of X in the operational series is reached, the work is finished.

Working with COM port

The block for working with the COM port is put into a separate unit, and the already used procedures are used in the main code. First of all, you need to initialize the COM port using the PortInit procedure, for this purpose the Win32 API creates a file and assigns the CommHandle variable to its handle:

CommHandle:= CreateFile(PChar(PortName),

GENERIC_READ or GNERIC_WRITE,0,nil,OPEN_EXISTING,

FILE_ATTRIBUTE_NORMAL or FILE_FLAG_OVERLAPPED,0);

Next, it is necessary to configure the port parameters, that is, the DCB structure, as well as the mask. The DCB structure is the control structure of the port. A mask is a description of an event that the port will wait for, and which will handle event processing.

```
//checking statement
  CheckError(CommHandle <> -1, 'Can not open '+ PortName +' port');
//setting the mask - " after the arrival of a certain symbol "
  SetCommMask(CommHandle,EV_RXFLAG);
  //initialization of DCB
   GetCommState(CommHandle,DCB);
   DCB.BaudRate:=CBR_9600;          //initialization of speed
   DCB.Parity:=NOPARITY;            //setting the absence of a parity check
   DCB.ByteSize:=8;                 //8 bits in the transmitted byte
   DCB.StopBits:=OneStopBit;        // Single stop bit
   DCB.EvtChar:=chr(13);             // assigning a symbol for the flag
   SetCommState(CommHandle,DCB);  //setting the DCB
```

Write to the port is implemented using a short procedure WriteComm, where the number of transmitted bytes is determined and the array Transmit: array [0 ...

255] of char is defined. In our case, only one byte of chr (A) is always transmitted, which is converted by the controller to an analog value: // send one character to the port

```
KolByte:=1;
Transmit[0]:=chr(A);
WriteFile(CommHandle,Transmit,KolByte,KolByte,@Ovr);.
```

After the completion of the main form, you must end the use of the port with the KillComm sub-routine, where CloseHandle (CommHandle) occurs; - "release" of the actual file-port.

4. DAC IMPLEMENTATION WITH MICROCONTROLLER ATmega328p

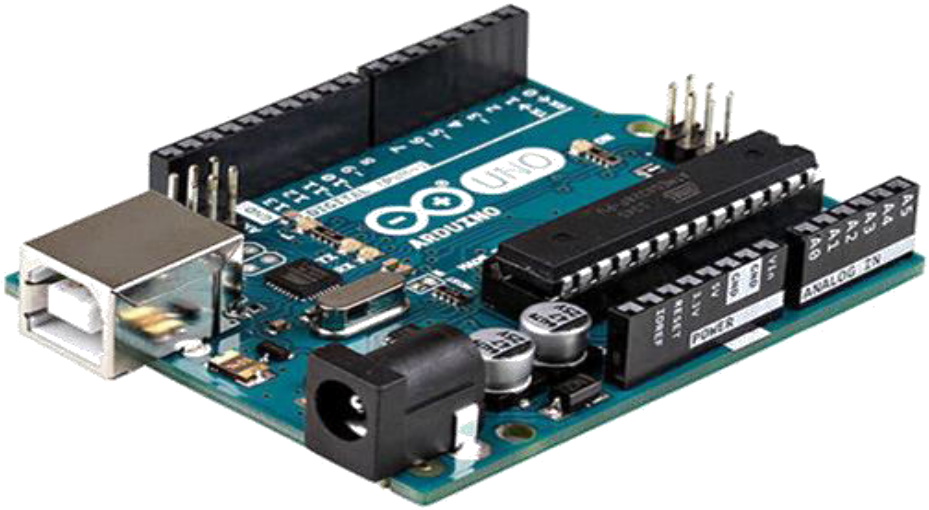

Fig. 6. Microcontroller Arduino UNO.

The sketch of the subroutine for the microcontroller is shown in Fig. 7.

The operation of the subroutine consists in transmitting the analog value (PWM wave) to the input / output port using the analogWrite () function.

The PWM signal frequency is approximately 490 Hz. Parameters of this function: pin - input / output port for which PWM signal is sent and value - cycle period value between 0 (completely off) and 255 (the signal is fed continuously). The port receives a value that comes from the PC via USB. On the platform Arduino Uno installed several devices to communicate with the computer, other devices Arduino or microcontrollers.

```
int G_Output = 0;
void setup() {
          // setup code, to run once:
  pinMode(5, OUTPUT);
  Serial.begin(9600);
}

void loop() {
          // main code, to run repeatedly:
   if (Serial.available())
   {      // If anything comes in Serial (USB),
     int a_In = Serial.read();
     if( G_Output != a_In )
     {
          //Set the PWM at G_Output % on digital pin 5
        G_Output = a_In;
        analogWrite(5, G_Output);
     }
   }
}
```

Fig. 7. Subroutine code.

ATmega328 supports a serial UART TTL (5 V) interface, carried out with the outputs 0 (RX) and 1 (TX). The ATmega8U2 chip installed on the board is controlled via the COM-USB interface, the program on the computer side cyclically sends a byte to the virtual COM port whose value is converted to the length of the PWM pulse. To work with ATmega8U2 in a Windows environment, it uses standard USB-COM drivers, no third-party drivers are required, but on Windows, the ArduinoUNO.inf file is required for connection. The ATmega328 microcontroller comes with a recorded bootloader, making it easy to record new programs without using external programmers. Communication is carried out by the original STK500 protocol. The average value of the PWM wave is the average value of the voltage supplied from the output for a period of time of the wave. A square wave shape with an amplitude of 0-5 V has an average value equal to the maximum amplitude (5 V) multiplied by the duty cycle. This means that you can change the average value of the PWM output using the analogWrite function. For example, if you set the operating cycle to 0% (analogWrite (pin, 0)), the

average value will be 0V, if you set it to 50% (analogWrite (pin, 127)), we get an average value of 2.5V, etc. Now, knowing how to change the average value of a wave, you need to use it. We need to output a constant current, which is the desired output of our DAC. We can extract the value of a constant signal using a low-pass filter. After that, the signal goes to an inverting amplifier with a gain of -5 to obtain the desired -25 VAs an op-amp, the microcircuit KR1408UD1 was used, which is a high-voltage operational amplifier with small input currents for constructing devices operating at elevated signal levels. The supply voltage of this opamp is ± 27 V. The schematic diagram of the ready amplifier and filter is shown in Fig. 8.

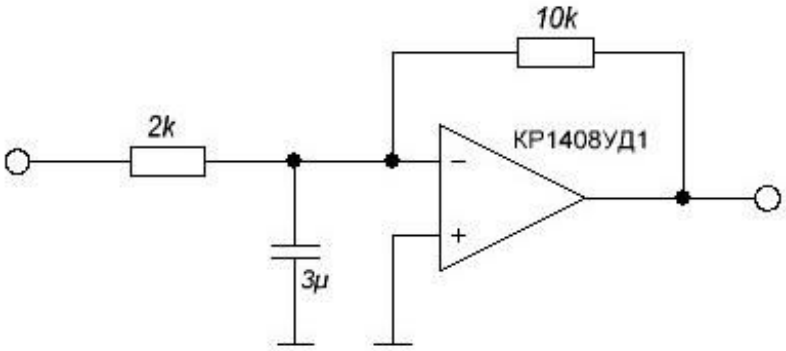

Fig. 8. Amplifier circuit

CONCLUSION

In the course of this research work, the task was solved, namely, a software and hardware system based on OS Windows and the Arduino UNO microcontroller was implemented. The practical use of the software and hardware complex demonstrated the validity of applying classical and mathematical interpolation methods for controlling a gas-discharge tube in a technological mode. However, it should be noted that there is a wide scope for improving the characteristics of this software and hardware system, which depends on the speed of the PC and the microcontroller used

REFERENCES

1. Kostrin DK, Ukhov AA Sensors in electronic devices. SPb .: Publishing house SPbGETU «LETI», 2013.
2. Kostrin D. K., Ukhov A. A. Microprocessor technology. Microprocessors and microcontrollers: Methodical instructions for conducting practical and laboratory exercises. SPb .: Publishing house SPbGETU «LETI», 2011.
3. Xavier Pacheco Delphi 6 Developer's Guide ISBN-10: 0-7686-5879-9 Published 2006 by Sams.