Mateusz Krawczak 241318

Karol Jaskółka 241306

Grupa: Pon P 17:00

Data wykonania ćwiczenia: 9.12.2019

Urządzenia Peryferyjne

Ćwiczenie 15 – Sterowaniem silnikiem krokowym za pomocą USB

1. Wstęp

Silnik krokowy jest silnikiem sterowanym dyskretnymi impulsami elektrycznymi. Taki sposób zasilania powoduje, że wirnik silnika może obrócić się o dowolnie zadany kąt. Dzięki temu kąt obrotu wirnika jest silnie zależny od liczby impulsów elektrycznych, którymi zasilamy nasz silnik.

Sterowanie urządzeniem obywa się poprzez bezpośrednie podłączenie (kablem na interfejs USB) silnika i komputera osobistego.

Wyróżnia się wiele rodzajów silników krokowych. Najczęstsze różnice występujące w nich to sposób sterowania nimi, liczba uzwojeń czy nawet rodzaj materiału, z którego zostały wykonane. Jednak można je podzielić na trzy większe grupy:

- Silniki VR – silniki o zmiennej reluktancji,
- Silniki PM i HB – silniki z magnesem stałym (PB) oraz hybrydowe (HB),
- Silniki bipolarne i unipolarne – silniki z dzielonymi uzwojeniami

Na zajęciach rozważanym silnikiem był silnik unipolarny, czyli taki silnik, który posiada tak zwane uzwojenia z odczepem, który pozwala na łatwiejszy sposób sterowania i ostatecznie na łatwiejszy sposób napisania oprogramowania (sterownika).

2. Zagadnienia

Możemy rozważać trzy tryby pracy silnika krokowego. Jest nim:

- **Tryb falowy** – w każdej fazie zasilania pracuje jedynie 25% wszystkich uzwojeń (tryb falowy to szczególny przypadek trybu pełnokrokowego).

- **Tryb pełnokrokowy** – zasilane są uzwojenia parami w odpowiedniej sekwencji (pokazanej poniżej).

- **Tryb półkrokowy –** jest to w pewnym sensie połączenie trybu falowego i pełnokrokowego. Naprzemiennie zasilane jest jedno oraz dwa uzwojenia. Dzięki takiej sekwencji zasilania bieguny wirnika silnika ustawiają się przeciwnie do biegunów stojana lub w połowie między nimi.

Silnik unipolarny składa się z dwóch uzwojeń. Oznacza to, że w przypadku sterowania półkrokowego musieliśmy „poduzwojenia” uzwojeń zasilać w odpowiednie sekwencji bitowej. Poniższa tabelka przedstawia ideę zasilania owego silnika:

0 1 0 1
0 0 0 1
1 0 0 1
1 0 0 0
1 0 1 0
0 0 1 0
0 1 1 0
0 1 0 0

W programie reprezentacja tych liczb binarnych została zapisana do tablicy bajtowej i przyjmuje ona postać:

```
byte[] LeftU = { 0x09, 0x08, 0x0A, 0x02, 0x06, 0x04, 0x05, 0x01 };
```

Dla pracy pełnokrokowej mamy mniej przypadków zasilania uzwojeń silnika. Wygląda to tak:

0	1	1	0
1	0	1	0
1	0	0	1
0	1	0	1

Programowa interpretacja takiego sposobu zasilania przedstawia się następująco:

```
byte[] LeftD = { 0x09, 0x0A, 0x06, 0x05 }; //dwufazowe krokowe
```

Ostatnim zaimplementowanym przez nas rodzajem pracy silnika krokowego jest praca falowa. Różni się ona od pracy pełnokrokowej tym, że wówczas jest wykorzystane 25% uzwojeń silnika (w przypadku pracy pełnokrokowej wykorzystujemy 50% uzwojeń).

0	1	0	0
0	0	1	0
1	0	0	0
0	0	0	1

```
byte[] LeftP = { 0x08, 0x02, 0x04, 0x01 };  // jednofazowe krokowe
```

Gdybyśmy chcieli obracać wirnikiem w drugą stronę, wystarczy wysyłać elementy tablic od końca.

3. Przebieg zajęć i kod programu

Do programu zostały zaimplementowane wyżej wymienione tablice. Do komunikacji naszego silnika przez interfejs USB posłużyliśmy się biblioteką **FTD2XX_NET**.

Cały program znajduję się pod tym linkiem

https://github.com/matson19/UP/tree/master/Lab%204%20-%20silnik%20korkowy

```
  public partial class Form1 : Form
    {

        FTD2XX_NET.FTDI.FT_STATUS ftstatus;
        FTD2XX_NET.FTDI.FT_DEVICE_INFO_NODE[] devicelist = new
FTD2XX_NET.FTDI.FT_DEVICE_INFO_NODE[1];
        FTDI device = new FTDI();

        byte[] LeftP = { 0x08, 0x02, 0x04, 0x01 };  // jednofazowe krokowe
        byte[] RightP = { 0x01, 0x04, 0x02, 0x08 };

        byte[] LeftD = { 0x09, 0x0A, 0x06, 0x05 }; //dwufazowe krokowe
        byte[] RightD = { 0x05, 0x06, 0x0A, 0x09 };

        byte[] LeftU = { 0x09, 0x08, 0x0A, 0x02, 0x06, 0x04, 0x05, 0x01 };  //
polkrokowe
        byte[] RightU = { 0x01, 0x05, 0x04, 0x06, 0x02, 0x0A, 0x08, 0x09 };

        byte[] stop = { 0x00 };

        int degree = 360;
        int speed = 50;

        public Form1()
        {
            InitializeComponent();
        }

        private void buttonLeft_Click(object sender, EventArgs e)
        {
            textBox.Text += "Obrot w lewo...\r\n";
            try
            {
                numericUpDownCounter.Enabled = false;
                rotate(LeftP, 4);
                numericUpDownCounter.Enabled = true;
                textBox.Text += "Obrot w lewo dziala\r\n";
            }
            catch(Exception ex)
            {
                textBox.Text += "Nie Dziala\r\n";
            }
        }

        private void buttonRight_Click(object sender, EventArgs e)
        {
            textBox.Text += "Obrot w prawo...\r\n";
            try
            {
```

```
            numericUpDownCounter.Enabled = false;
            rotate(RightP, 4);
            numericUpDownCounter.Enabled = true;
            textBox.Text += "Obrot w prawo dziala\r\n";
        }
        catch (Exception ex)
        {
            textBox.Text += "Nie Dziala\r\n";
        }
    }

    private void buttonRight2_Click(object sender, EventArgs e)
    {
        textBox.Text += "Obrot w prawo dwufazowy...\r\n";
        try
        {
            numericUpDownCounter.Enabled = false;
            rotate(RightD, 4);
            numericUpDownCounter.Enabled = true;
            textBox.Text += "Obrot w prawo dwufazowy dziala\r\n";
        }
        catch (Exception ex)
        {
            textBox.Text += "Nie Dziala\r\n";
        }
    }

    private void buttonLeft2_Click(object sender, EventArgs e)
    {
        textBox.Text += "Obrot w lewo dwufazowy...\r\n";
        try
        {
            numericUpDownCounter.Enabled = false;
            rotate(LeftD, 4);
            numericUpDownCounter.Enabled = true;
            textBox.Text += "Obrot w lewo dwufazowy dziala\r\n";
        }
        catch (Exception ex)
        {
            textBox.Text += "Nie Dziala\r\n";
        }
    }

    private void buttonRightHalf_Click(object sender, EventArgs e)
    {
        textBox.Text += "Obrot w prawo polkrokowy...\r\n";
        try
        {
            numericUpDownCounter.Enabled = false;
            rotate(RightU, 8);
            numericUpDownCounter.Enabled = true;
            textBox.Text += "Obrot w prawo polkrokowy dziala\r\n";
        }
        catch (Exception ex)
        {
            textBox.Text += "Nie Dziala\r\n";
        }
    }

    private void buttonLeftHalf_Click(object sender, EventArgs e)
    {
        textBox.Text += "Obrot w lewo polkrokowy...\r\n";
```

```
            try
            {
                numericUpDownCounter.Enabled = false;
                rotate(LeftU, 8);
                numericUpDownCounter.Enabled = true;
                textBox.Text += "Obrot w lewo polkrokowy dziala\r\n";
            }
            catch (Exception ex)
            {
                textBox.Text += "Nie Dziala\r\n";
            }
        }

        public void rotate(byte[] buff, int steps)
        {
            int loops = (degree * 10) / 74;

            if (steps == 8)
            {
                loops *= 2;
            }

            Int32 bytesToWrite = 1;
            UInt32 bytesWritten = 0;

            byte[][] controlByte = new byte[steps][];

            for (int i = 0; i < steps; i++)
                controlByte[i] = new byte[] { buff[i] };

            for (int k = 0, i = steps - 1; k < loops; k++)
            {

                ftstatus = device.Write(controlByte[i], bytesToWrite, ref
bytesWritten); //przesłanie odpowiedniej kombinacji bitowej do sterownika
                System.Threading.Thread.Sleep(1000 / speed);

                if (i == 0)
                    i = steps - 1;
                else
                    i--;
            }

            // wysłanie 0x00 w celu zaprzestanie wysylania stanów wysokich do silnika
krokowego
            ftstatus = device.Write(stop, 1, ref bytesWritten);
        }

        private void buttonConnect_Click(object sender, EventArgs e)
        {
            ftstatus = device.GetDeviceList(devicelist);
            try
            {
                ftstatus = device.OpenByDescription(devicelist[0].Description);
                ftstatus = device.SetBitMode(0xff, 1); //pozwala na przesylanie danych
do ukladu
                textBox.Text += "Status Polaczenia: " + ftstatus.ToString() + "\r\n";
            }

            catch (Exception ex)
            {
                textBox.Text += "Status Polaczenia: FT_FAIL \r\n";
```

```
            }
        }

        private void numericUpDownCounter_ValueChanged(object sender, EventArgs e)
        {
            speed = Convert.ToInt32(numericUpDownCounter.Value);
        }

        private void numericUpDownDegree_ValueChanged(object sender, EventArgs e)
        {
            degree = Convert.ToInt32(numericUpDownDegree.Value);
        }


    }
```

4. Podsumowanie

Program działał poprawnie, zostały wykonane wszystkie zadania z instrukcji.