Urządzenia Peryferyjne

Silnik Krokowy

Nikita Stepanenko, 245816

Termin zajęć Środa 11:00 TP

1.Cel ćwiczenia

Celem wykonywanego ćwiczenie było:
1) Zapoznać się z właściwościami systemowymi urządzenia USB sterującego silnikiem.

2) Uruchomić przykładowy program do obsługi sterownika i przetestować jego możliwości.

3) Napisać aplikację, która będzie umożliwiać:

a) otwarcie i zamknięcie urządzenia (może to się odbywać automatycznie na stracie i przy wyjściu z program);
b) wybór sposobu sterowania silnikiem: falowe, pełnokrokowe, półkrokowe;
c) wykonanie n kroków w prawo i w lewo (najlepiej wykonywanie n kroków w prawo i w lewo wykonać za pomocą dwóch przycisków. Jednokrotne naciśnięcie przycisku ma spowodować wykonanie n kroków);
d) wprowadzenie liczby kroków n oraz czas trwania jednego kroku.

2.Wstęp teoretyczny

Silnik krokowy – silnik elektryczny, w którym impulsowe zasilanie prądem elektrycznym powoduje, że jego wirnik nie obraca się ruchem ciągłym, lecz wykonuje za każdym razem ruch obrotowy o ściśle ustalony kąt. Dzięki temu kąt obrotu wirnika jest ściśle zależny od liczby dostarczonych impulsów prądowych, a prędkość kątowa wirnika jest dokładnie równa częstotliwości impulsów pomnożonej przez wartość kąta obrotu wirnika w jednym cyklu pracy silnika.

Istnieją różne rodzaje silników krokowych. Najczęstsze różnice występujące w nich to sposób sterowania nimi, liczba uzwojeń czy nawet rodzaj materiału, z którego zostały wykonane. Można je podzielić na trzy główne rodzaje:

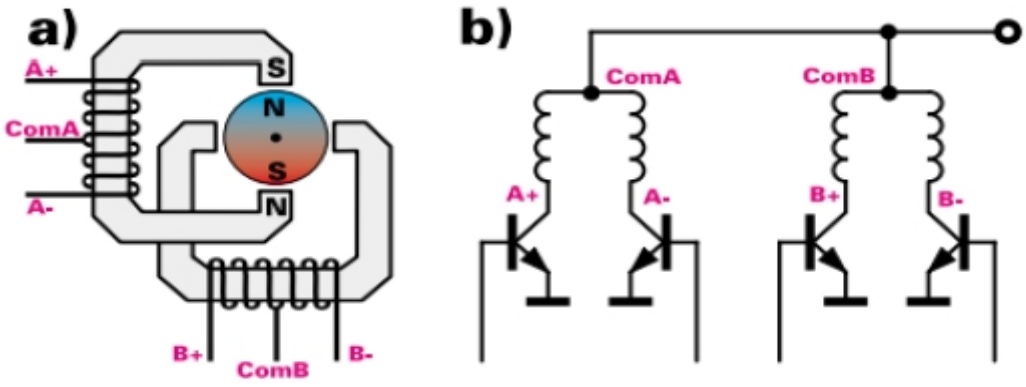
1. Silniki VR – silniki o zmiennej reluktancji,
2. Silniki PM i HB – silniki z magnesem stałym (PB) oraz hybrydowe (HB),
3. Silniki bipolarne i unipolarne – silniki z dzielonymi uzwojeniami

Rozważamy trzy tryby pracy silnika krokowego:

- **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
- **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.

W czasie zajęć pracowaliśmy z silnikiem unipolarnym. Unipolarny silnik posiada uzwojenie z odczepem. Zapoznaliśmy się z jego budową oraz działaniem na stronie https://elportal.pl/pdf/k01/81_24.pdf

Na tej stronie zapoznaliśmy się ze schematem silnika unipolarnego:



Silnik unipolarny składa się z dwóch uzwojeń. Oznacza to, że dla różnych typów sterowania musieliśmy „poduzwojenia" uzwojeń zasilać w odpowiednie sekwencji bitowej.

Dla sterowania półkrokowego mamy następną ideę zasilania owego silnika:

0010
0110
0100
0101
0001
1001
1000
1010

W programie reprezentacja tych liczb binarnych została zapisana do tablicy bajtowej.

Ten tabela w postaci Heksadecymalnej `0x02, 0x06, 0x04, 0x05, 0x01, 0x09, 0x08, 0x0A`

Dla sterowania pełnokrowego mamy następną ideę zasilania owego silnika:

0110
0101
1001
1010

Ten tabela w postaci Heksadecymalnej `0x06, 0x05, 0x09, 0x0A`

Dla sterowania falowego mamy taką samą ideę zasilania owego silnika jak i dla pełnokrokowego:

0010
0100
0001
1000

Ten tabela w postaci Heksadecymalnej 0x02, 0x04, 0x01, 0x08

Jedyna różnica falowego od pełnokrokowego w tym, że pełnokrokowy korzysta 50% uzwojeń silnika, a falowe – 25%.

Dla obracania w inną stronę trzeba po prostu wysyłać elementy tablic od końca.

3.Realizacja ćwiczenia

W trakcie zajęć udało się wykonać następne zadania:

1) Zapoznać się z właściwościami systemowymi urządzenia USB sterującego silnikiem.

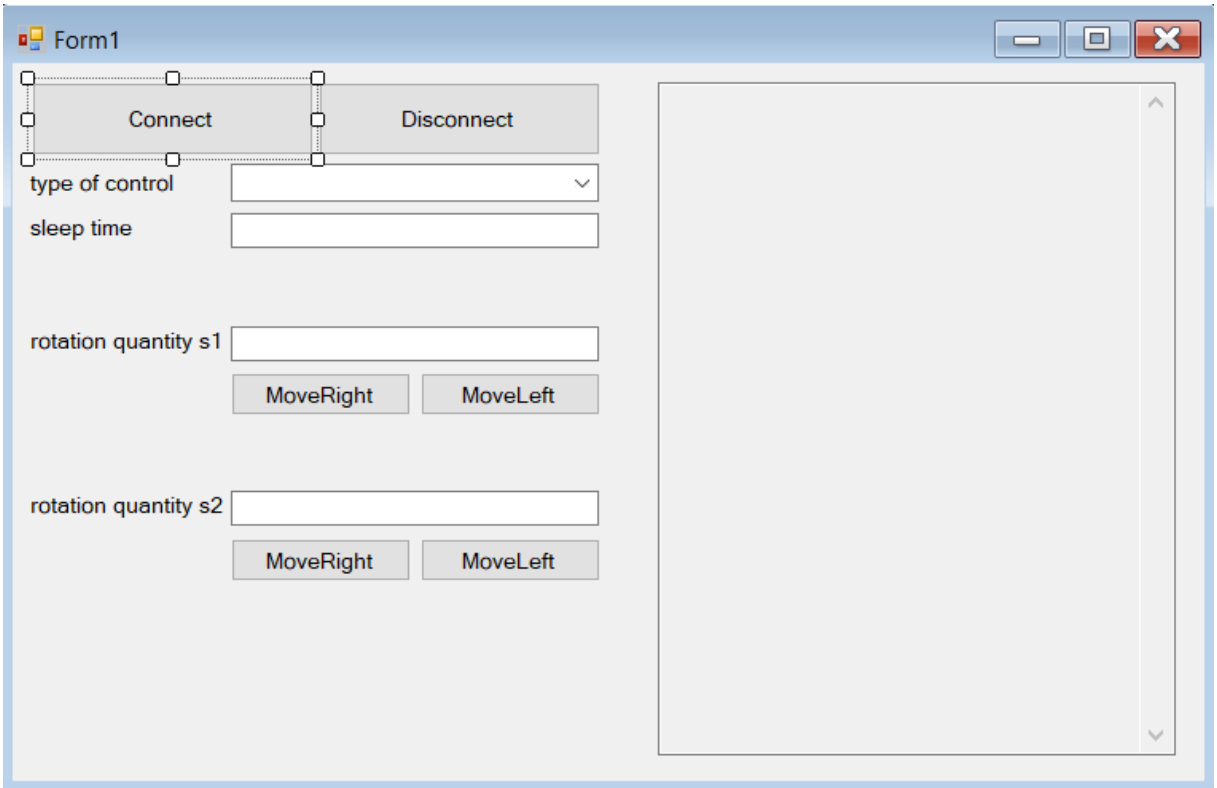
2) Uruchomić przykładowy program do obsługi sterownika i przetestować jego możliwości.

3) Napisać aplikację, która będzie umożliwiać:

a) otwarcie i zamknięcie urządzenia (może to się odbywać automatycznie na stracie i przy wyjściu z program);
b) wybór sposobu sterowania silnikiem: falowe, pełnokrokowe, półkrokowe;
c) wykonanie n kroków w prawo i w lewo (najlepiej wykonywanie n kroków w prawo i w lewo wykonać za pomocą dwóch przycisków. Jednokrotne naciśnięcie przycisku ma spowodować wykonanie n kroków);
d) wprowadzenie liczby kroków n oraz czas trwania jednego kroku.

Opis programu

Główne okno programu:



Wykorzystane biblioteki

```
using System;
using System.Windows.Forms;
using FTD2XX_NET;
using System.Threading;
```

Zmienne globalne:

Tablicy bajtowe słurzą do zasilania pewnego uzwojenia co powoduje powstanie pola magnetycznego i rotar powraca się.

```
FTDI deviceFTDI; // definiowanie urządzenia
FTDI.FT_STATUS status; // status urządzenia

byte[] stopMotion = { 0x00}; // odłącenie zasilania
// zaslanie dla silnika 1
byte[] halfMotion = { 0x02, 0x06, 0x04, 0x05, 0x01, 0x09, 0x08, 0x0A }; // tablica zasilania sterowania półkrokowego
byte[] fullMotion = { 0x06, 0x05, 0x09, 0x0A }; // tablica zasilania sterowania pełnokrowego
byte[] waveMotion = { 0x02, 0x04, 0x01, 0x08 };// tablica zasilania sterowania falowego
// zaslanie dla silnika 2
byte[] halfMotion2 = { 0x20, 0x60, 0x40, 0x50, 0x10, 0x90, 0x80, 0xA0 }; // tablica zasilania sterowania półkrokowego
byte[] fullMotion2 = { 0x60, 0x50, 0x90, 0xA0 };// tablica zasilania sterowania pełnokrowego
byte[] waveMotion2 = { 0x20, 0x40, 0x10, 0x80 };// tablica zasilania sterowania falowego

int idx1 = 0; //pozycja ineratora w tablicy zasilania (silnik 1)
int idx2 = 0; //pozycja ineratora w tablicy zasilania (silnik 2)
```

Po nacisnięciu przycisku „**Connect**” skanujemy urządzenia FTDI i wybieramy 1-y z listy, ustalamy tryb działania i sprawdzamy: czy jest połączenie.

```
    private void buttonConnect_Click(object sender, EventArgs e)
    {
        try
        {
            // liczba urządzen = 0
            UInt32 countDeviceFTDI = 0;

            deviceFTDI = new FTDI();
            // skanowanie urządzeń
            deviceFTDI.GetNumberOfDevices(ref countDeviceFTDI);
            FTDI.FT_DEVICE_INFO_NODE[] deviceList = new FTDI.FT_DEVICE_INFO_NODE[countDeviceFTDI];
            deviceFTDI.GetDeviceList(deviceList);
            // wybierany jest 1-y z listy urządzeń
            status = deviceFTDI.OpenBySerialNumber(deviceList[0].SerialNumber);
            // ustalienie trybu
            deviceFTDI.SetBitMode(0xff, 1);
            if(status == FTDI.FT_STATUS.FT_OK)
            {
                textBox1.Text += "Connected" + Environment.NewLine;
            }
            else
            {
                textBox1.Text += status.ToString() + Environment.NewLine;
            }
        }
        catch(Exception ex)
        {
            textBox1.Text += ex.Message + Environment.NewLine;
        }

    }
```

Metoda sterująca obracaniem rotera:

Dane wchodzące do metody:

Tab – tablica zawierająca kody bajtowe oznaczające które uzwojenie trzeba włączyć.

Steps – ilość iteracji pętli = ilość kroków rotera

Time – czas oczekiwania między krokami

Left – kierunek obracania (-1 - jedna strona, 1 - inna)

Idx – ostatnia pozycja w tablicy

```
13 references
void stepMotion(byte[] tab, int steps, int time, int left, ref int idx)
{
    int bytesToWrite = 1;
    uint bytesWritten = 0;
    for(int i = 0; i< steps; i++)
    {
        idx += left;
        if(idx > tab.Length-1)
        {
            idx = 0;
        }
        if(idx < 0)
        {
            idx = tab.Length - 1;
        }
        byte[] currentByte = { tab[idx] };
        deviceFTDI.Write(currentByte, bytesToWrite, ref bytesWritten);
        Thread.Sleep(time);
    }

}
```

Dla różnych metod sterowania silnikiem, po prostu wysyłam do metody `stepMotion` różne tablicy sterowania (look – zmienne globalne na górze)

Nie będę opisywał obracanie w lewo i w prawo dla obu silników, bo rórznica polega tylko na tym, którą tablicę bajtową wysyłam do metody (silnik1 lub 2), 1 lub -1 wysyłam do zmiennej ”left”(obracanie się w lewo lub w prawo) oraz idx(musi być zdodny z odpowiednią tablicą bajtową). Zmienne do wymiany oznacowe w komientarzu do kodu.

```
1 reference
private void buttonMoveRight_Click(object sender, EventArgs e)
{

    switch (comboBoxEngine.SelectedIndex)
    {
        case 0:
            {//              tab                                                                                left     idx
                stepMotion(halfMotion, System.Convert.ToInt32(textBoxSteps.Text), System.Convert.ToInt32(textBoxStepTime.Text), 1, ref idx1);
                break;
            }
        case 1:
            {
                stepMotion(fullMotion, System.Convert.ToInt32(textBoxSteps.Text), System.Convert.ToInt32(textBoxStepTime.Text), 1, ref idx1);
                break;
            }
        case 2:
            {
                stepMotion(waveMotion, System.Convert.ToInt32(textBoxSteps.Text), System.Convert.ToInt32(textBoxStepTime.Text), 1, ref idx1);
                break;
            }
    }
}
```

4.Wnioski

To ćwiczenie nie było trudne do wykonania. Przygotowywanie do tego ćwiczenia zajęło dużo czasu, bo trzeba było przeczytać sporo dokumentacji. Zrozumienie tego, jaka pracuje silnik krokowy nie było trudnym. Podczas napisania programu, trochę problemu było z tym, żeby przy zmianie kierunku obrotu, silnik od razu zaczynał obracać w inną stron oraz nie było żadnych dziwnych ruchów. Ale udało się to zrobić. Wszystko się działało poprawnie.