```
using System;
using System.Net;
using System.Net.Sockets;
using System.Threading;
using Microsoft.SPOT;
using Microsoft.SPOT.Hardware;
using SecretLabs.NETMF.Hardware;
using SecretLabs.NETMF.Hardware.Netduino;
using static System.Math;

namespace Greesha
{
    enum Calibration: int
    {
        north = 2420,
        south = 3150,
        east = 1900,
        west = 2300,
    }

    class Robot
    {
        InputPort button = new InputPort(Pins.ONBOARD_BTN, true,
          Port.ResistorMode.PullUp);
        PWM leftFrontMotor = new PWM(PWMChannels.PWM_PIN_D11, 1000, 0, false);
        PWM leftBackMotor = new PWM(PWMChannels.PWM_PIN_D6, 1000, 0, false);
        PWM rightFrontMotor = new PWM(PWMChannels.PWM_PIN_D10, 1000, 0, false);
        PWM rightBackMotor = new PWM(PWMChannels.PWM_PIN_D9, 1000, 0, false);

        bool check = false;


        public void Go(int degree, int time)
        {
            Compass Magnetometer = new Compass();

            byte[] xyz = new byte[6];
            double delta, x, y, turn, right, left, forsage;
            int finishTime, timer = 0;

            finishTime = DateTime.Now.Minute * 60 + DateTime.Now.Second + time;

            do
            {
                // Считываем данные с 3х осевого компаса I2C
                xyz = Magnetometer.Read();
                // Преобразуем 2 байта с координаты в переменную
                x = xyz[1] * 255 + xyz[0];
                y = xyz[3] * 255 + xyz[2];

                delta = degree - CurrentDegree(x, y);
                // Разность заданного и текущего углов движения
                turn = Tan((delta * PI / 180) / 2) / 2;
                // "-0.5" - поворот направо, "+0.5" - влево
```

```
                if ( Abs(turn) > 0.5 )
                {
                    turn = 0.5 * Abs(turn) / turn;
                    finishTime = DateTime.Now.Minute * 60 + DateTime.Now.Second +
                     time;
                }
                else
                {
                    timer = DateTime.Now.Minute * 60 + DateTime.Now.Second;
                }

                right = 0.5 - turn;
                left = 0.5 + turn;


                // Увеличиваем скорость до максимума
                // с сохранением крутящего момента
                forsage = 1 - Max(left, right);
                left += forsage;
                right += forsage;

                Motors(left, right);

            } while (timer < finishTime);

        }



        private double CurrentDegree(double x, double y)
        {
            double degree = 0.000000001;

            degree = 180 * (x - (double)Calibration.north) /
               ((double)Calibration.south - (double)Calibration.north);

            if (y < ((double)Calibration.west + (double)Calibration.east) / 2)
                degree = 360 - degree;

            return degree;
        }


        public void Motors(double left, double right)
        {
            leftFrontMotor.DutyCycle = left;
            leftBackMotor.DutyCycle = left;
            rightFrontMotor.DutyCycle = right;
            rightBackMotor.DutyCycle = right;

            if (button.Read())
            {
                if (check == false)
                {
                    leftFrontMotor.Start();
                    leftBackMotor.Start();
```

```
                    rightFrontMotor.Start();
                    rightBackMotor.Start();
                    check = true;
                }
                else
                {
                    leftFrontMotor.Stop();
                    leftBackMotor.Stop();
                    rightFrontMotor.Stop();
                    rightBackMotor.Stop();
                    check = false;
                }
            }


        }

    }




    class Compass
    {
        static I2CDevice.Configuration config = new I2CDevice.Configuration(28,
          100);
        static I2CDevice compass = new I2CDevice(config);


        public Compass()
        {
            byte[] settings1 = new byte[] { 0x20, 125 };
            byte[] settings2 = new byte[] { 0x23, 12 };
            WriteToCompass(settings1);
            WriteToCompass(settings2);
        }


        public byte[] Read()
        {
            byte[] reboot = new byte[] { 0x21, 120 };
            byte[] coordinates = new byte[] { 0x28 };
            byte[] XYZ = new byte[6];

            WriteToCompass(coordinates);
            XYZ = ReadFromCompass();
            WriteToCompass(reboot);

            return XYZ;
        }


        public void WriteToCompass(byte[] command)
        {
            I2CDevice.I2CWriteTransaction writeTransaction =
              I2CDevice.CreateWriteTransaction(command);
```

```
            I2CDevice.I2CTransaction[] transaction = new I2CDevice.I2CTransaction[]
               { writeTransaction };

            compass.Execute(transaction, 1000);
            compass.Execute(transaction, 1000);
        }


        public byte[] ReadFromCompass()
        {
            byte[] inBuffer = new byte[6];
            I2CDevice.I2CReadTransaction readTransaction =
              I2CDevice.CreateReadTransaction(inBuffer);
            I2CDevice.I2CTransaction[] transaction = new I2CDevice.I2CTransaction[]
               { readTransaction };

            compass.Execute(transaction, 1000);
            return inBuffer;
        }
    }
}
```