ภาคผนวก

ภาคผนวก ก
โปรแกรมควบคุมหุ่นยนต์เดลต้า

โปรแกรมควบคุมหุ่นยนต์เดลต้า

```
// ประกาศ namespace ที่จำเป็นต้องใช้
using NationalInstruments;
using NationalInstruments.UI;
using NationalInstruments.DAQmx;
using NationalInstruments.UI.WindowsForms;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Threading;
using System.Net;
using System.Net.Sockets;
using System.Collections;
namespace WindowsApplication20
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        // เมื่อมีเหตุการณ์ โหลดหน้าฟอร์มขึ้นมา

        private void Form1_Load(object sender, EventArgs e)
        {
```

```
            // สร้างอินสแทนซ์จากคลาสต้นแบบ
            _Pulse = new Pulse();   // สร้างอินสแทนซ์ Pulse Output
            _AnalogOut = new AnalogOut(new double[3] { 2, 2, 2 });  // สร้างอินสแทนซ์
Analog Output
            _kinematics = new kinematics(346.410161513775458705489268301 17,
155.884572681198956417470170735 53, 180.0, 450.0);   // สร้างอินสแทนซ์ในการคำนวน
kinematics
            _Digital = new DigitalOut(0xE0E0E000);  // สร้างอินสแทนซ์ Digital Output
            txt_IP.Text = GetIP();  // หา IP ของเครื่อง
        }
        // เมื่อมีเหตุการณ์ ปิดหน้าพอร์ม
        private void Form1_FormClosing(object sender, FormClosingEventArgs e)
        {
            CloseSockets(); // ปิดพอร์ตในการสร้างเซิร์ฟเวอร์
            if (_ReadEncoder != null)   // อ่านเอนโค้ดเดอร์อยู่หรือไม่
            {
                _ReadEncoder.Disable();     // หยุดอ่านเอนโค้ดเดอร์
            }
            _Digital.DigitalWrite(0xE0E0E000);  // motor off
        }
        #region Dalta Robot
        // ประกาศตัวแปลอินสแทนซ์จากคลาสต้นแบบ
        ReadEncoder _ReadEncoder;
        DigitalOut _Digital;
        kinematics _kinematics;
        Pulse _Pulse;
        AnalogOut _AnalogOut;
        // โชว์มุมและตำแหน่ง
        public void showEncoder(MotorAngle value)
        {
```

```
            double[] position = _kinematics.calcForward(value[0], value[1], value[2]);  //
นำองศามาคำนวนหาตำแหน่งของทูล
            // ใช้อับเดทค่าแบบ asynchronously
            this.BeginInvoke(new EventHandler(delegate
            {
                numericEditArray1.SetValues(new double[3] { value.AngleM1,
value.AngleM2, value.AngleM3 }, 0, 3);   // แสดงค่ามุม
                numericEditArray2.SetValues(new double[3] { position[1], position[2],
position[3] }, 0, 3);          // แสดงค่าตำแหน่งของทูล
            }));
        }
        // สวิตส์ on-off
        private void switch1_StateChanged(object sender, ActionEventArgs e)
        {
            if (switch1.Value)      // เปิดสวิตส์หรือไม่
            {
                _Digital.DigitalWrite(0xE5E5E500);      // motor on
                Thread.Sleep(100);      // รอ 0.1 วินาที
                _ReadEncoder = new ReadEncoder(showEncoder, 10, true);  // สร้างอินส
แทนซ์ Read Encoder
                _ReadEncoder.Enable();  // เริ่มอ่าน encoder
                Thread.Sleep(100);      // รอ 0.1 วินาที
                HomeWithindex();    // สั่งหา Home ของหุ่นยนต์
            }
            else
            {
                _Digital.DigitalWrite(0xE0E0E000);      // motor off
                if (_ReadEncoder != null)       // อ่านเอนโค้ดเดอร์อยู่หรือไม่
                {
                    _ReadEncoder.Disable();     // หยุดอ่านเอนโค้ดเดอร์
```

```
            }
        }
    }
    // สั่งองศามอเตอร์
    private void btn_setangle_Click(object sender, EventArgs e)
    {
        // สั่งให้มอเตอร์ไปตามองศาที่กำหนดโดยมีความถี่ของสัญญาณ 500kHz
        _Pulse.PulseGeneration(-numericEdit1.Value, 500000.0, -numericEdit1.Value,
500000.0, -numericEdit1.Value, 500000.0);
    }
    #region CreatePulsehome
    private delegate void _DelegatePulsehome();     // สร้างตัวแปล delegate ชื่อ
_DelegatePulsehome
    // หาตำแหน่ง home robot
    private void HomeWithindex()
    {
        _DelegatePulsehome DelegatePulsehome1 = new
_DelegatePulsehome(CreatePulsehome1);   // สร้างอินสแทนซ์ในการหา home ของมอเตอร์
ตัวที่ 1
        IAsyncResult Homewinder1 = DelegatePulsehome1.BeginInvoke(null, null);
// เริ่มการทำงานแบบ asynchronously เพื่อหา home ของมอเตอร์ตัวที่ 1
        _DelegatePulsehome DelegatePulsehome2 = new
_DelegatePulsehome(CreatePulsehome2);   // สร้างอินสแทนซ์ในการหา home ของมอเตอร์
ตัวที่ 2
        IAsyncResult Homewinder2 = DelegatePulsehome2.BeginInvoke(null, null);
// เริ่มการทำงานแบบ asynchronously เพื่อหา home ของมอเตอร์ตัวที่ 2
        _DelegatePulsehome DelegatePulsehome3 = new
_DelegatePulsehome(CreatePulsehome3);   // สร้างอินสแทนซ์ในการหา home ของมอเตอร์
ตัวที่ 3
        IAsyncResult Homewinder3 = DelegatePulsehome3.BeginInvoke(null, null);
// เริ่มการทำงานแบบ asynchronously เพื่อหา home ของมอเตอร์ตัวที่ 2
```

```
        }
        // หาตำแหน่ง home motor1
        public void CreatePulsehome1()
        {
            try
            {
                // กำหนดทิศทางให้มอเตอร์หมุนขึ้น
                using (Task digitalWriteTask = new Task())
                {
                    digitalWriteTask.DOChannels.CreateChannel("Dev1/port0/line0",
"DigitalOut_0", ChannelLineGrouping.OneChannelForEachLine);   // กำหนดขาของบอร์ดที่
จะใช้
                    DigitalSingleChannelWriter writer = new
DigitalSingleChannelWriter(digitalWriteTask.Stream);    // สั่งให้เตรียมพร้อมทำงาน
                    writer.WriteSingleSampleSingleLine(true, false);    // สั้งให้ส่งค่า false หรือ
ให้ลอจิกเป็น 0
                }
                // ให้เริ่มจ่ายสัยญาณให้มอเอตร์หมุน
                using (Task myTask = new Task())
                {
                    myTask.COChannels.CreatePulseChannelTime("Dev1/ctr0",
"PulseOutpu1", COPulseTimeUnits.Seconds, COPulseIdleState.Low, 0, 5E-05, 5E-05);
// กำหนดขาของบอร์ดที่จะใช้

myTask.Timing.ConfigureImplicit(SampleQuantityMode.ContinuousSamples, 50);  // สั้ง
ให้เตรียมพร้อมทำงาน
                    myTask.Start(); //ให้เริ่มจ่ายสัญญาณ
                    do
                    {
```

```
                    if (numericEditArray1.GetValues()[0] < 0) myTask.Stop();    // ถ้าถึงที่
ต้องการให้หยุดจ่ายสัญญาณ
                } while (!myTask.IsDone);
            }
            Thread.Sleep(1500);     // หยุดรอ 1.5 วินาที
            double buffpluse = numericEditArray1.GetValues()[0];    // อ่านค่าตำแหน่ง
ของมอเตอร์ตัวที่ 1
            // ถ้าองศาน้อยกว่า 0 ให้กำหนดทิศทางให้มอเตอร์หมุนลง
            if (buffpluse < 0)
            {
                using (Task digitalWriteTask = new Task())
                {
                    digitalWriteTask.DOChannels.CreateChannel("Dev1/port0/line0",
"DigitalOut_0", ChannelLineGrouping.OneChannelForEachLine);
                    DigitalSingleChannelWriter writer = new
DigitalSingleChannelWriter(digitalWriteTask.Stream);
                    writer.WriteSingleSampleSingleLine(true, true);
                }
            }
            // ถ้าองศามากกว่า 0 ให้กำหนดทิศทางให้มอเตอร์หมุนขึ้น
            else if (buffpluse > 0)
            {
                using (Task digitalWriteTask = new Task())
                {
                    digitalWriteTask.DOChannels.CreateChannel("Dev1/port0/line0",
"DigitalOut_0", ChannelLineGrouping.OneChannelForEachLine);
                    DigitalSingleChannelWriter writer = new
DigitalSingleChannelWriter(digitalWriteTask.Stream);
                    writer.WriteSingleSampleSingleLine(true, false);
                }
            }
```

```
            // ถ้าค่าไม่เท่ากับ 0 ให้มอเตอร์หมุดไปที่ตำแหน่ง Home
            if (buffpluse != 0)
            {
                using (Task myTask = new Task())
                {
                    myTask.COChannels.CreatePulseChannelFrequency("Dev1/ctr0",
"PulseTrain1", COPulseFrequencyUnits.Hertz, COPulseIdleState.Low, 0.0, 10000.0, 0.5);

myTask.Timing.ConfigureImplicit(SampleQuantityMode.FiniteSamples,
Convert.ToInt32(Math.Abs(buffpluse) * 262144 / 360));
                    myTask.Start();
                    do
                    {

                    } while (!myTask.IsDone);   // เช็คว่าจ่ายสัญญาณเสร็จยัง
                }
            }
        }
        catch
        {
        }
    }

    // หาตำแหน่ง home motor2
    public void CreatePulsehome2()
    {
        try
        {
            using (Task digitalWriteTask = new Task())
            {
```

```
                digitalWriteTask.DOChannels.CreateChannel("Dev1/port0/line2",
"DigitalOut_2", ChannelLineGrouping.OneChannelForEachLine);
                DigitalSingleChannelWriter writer = new
DigitalSingleChannelWriter(digitalWriteTask.Stream);
                writer.WriteSingleSampleSingleLine(true, false);
            }
            using (Task myTask = new Task())
            {
                myTask.COChannels.CreatePulseChannelTime("Dev1/ctr2",
"PulseOutput2", COPulseTimeUnits.Seconds, COPulseIdleState.Low, 0, 5E-05, 5E-05);

myTask.Timing.ConfigureImplicit(SampleQuantityMode.ContinuousSamples, 50);
                myTask.Start();
                do
                {
                    if (numericEditArray1.GetValues()[1] < 0) myTask.Stop();
                } while (!myTask.IsDone);
            }
            Thread.Sleep(1500);
            double buffpluse = numericEditArray1.GetValues()[1];
            if (buffpluse < 0)
            {
                using (Task digitalWriteTask = new Task())
                {
                    digitalWriteTask.DOChannels.CreateChannel("Dev1/port0/line2",
"DigitalOut_2", ChannelLineGrouping.OneChannelForEachLine);
                    DigitalSingleChannelWriter writer = new
DigitalSingleChannelWriter(digitalWriteTask.Stream);
                    writer.WriteSingleSampleSingleLine(true, true);
                }
            }
```

```
            else if (buffpluse > 0)
            {
                using (Task digitalWriteTask = new Task())
                {
                    digitalWriteTask.DOChannels.CreateChannel("Dev1/port0/line2",
"DigitalOut_2", ChannelLineGrouping.OneChannelForEachLine);
                    DigitalSingleChannelWriter writer = new
DigitalSingleChannelWriter(digitalWriteTask.Stream);
                    writer.WriteSingleSampleSingleLine(true, false);
                }
            }
            if (buffpluse != 0)
            {
                using (Task myTask = new Task())
                {
                    myTask.COChannels.CreatePulseChannelFrequency("Dev1/ctr2",
"PulseTrain2", COPulseFrequencyUnits.Hertz, COPulseIdleState.Low, 0.0, 10000.0, 0.5);
                   myTask.Timing.ConfigureImplicit(SampleQuantityMode.FiniteSamples,
Convert.ToInt32(Math.Abs(buffpluse) * 262144 / 360));
                    myTask.Start();
                    do
                    {
                    } while (!myTask.IsDone);
                }
            }
        }
        catch
        {
        }
    }
```

```
        // หาตำแหน่ง home motor3
        public void CreatePulsehome3()
        {
            try
            {
                using (Task digitalWriteTask = new Task())
                {
                    digitalWriteTask.DOChannels.CreateChannel("Dev1/port0/line1",
"DigitalOut_3", ChannelLineGrouping.OneChannelForEachLine);
                    DigitalSingleChannelWriter writer = new
DigitalSingleChannelWriter(digitalWriteTask.Stream);
                    writer.WriteSingleSampleSingleLine(true, false);
                }
                using (Task myTask = new Task())
                {
                    myTask.COChannels.CreatePulseChannelTime("Dev1/ctr1",
"PulseOutput3", COPulseTimeUnits.Seconds, COPulseIdleState.Low, 0, 5E-05, 5E-05);

myTask.Timing.ConfigureImplicit(SampleQuantityMode.ContinuousSamples, 50);
                    myTask.Start();
                    do
                    {
                        if (numericEditArray1.GetValues()[2] < 0) myTask.Stop();
                    } while (!myTask.IsDone);
                }
                Thread.Sleep(1500);
                double buffpluse = numericEditArray1.GetValues()[2];
                if (buffpluse < 0)
                {
                    using (Task digitalWriteTask = new Task())
```

```
                {
                    digitalWriteTask.DOChannels.CreateChannel("Dev1/port0/line1",
"DigitalOut_3", ChannelLineGrouping.OneChannelForEachLine);
                    DigitalSingleChannelWriter writer = new
DigitalSingleChannelWriter(digitalWriteTask.Stream);
                    writer.WriteSingleSampleSingleLine(true, true);
                }
            }
            else if (buffpluse > 0)
            {
                using (Task digitalWriteTask = new Task())
                {
                    digitalWriteTask.DOChannels.CreateChannel("Dev1/port0/line1",
"DigitalOut_3", ChannelLineGrouping.OneChannelForEachLine);
                    DigitalSingleChannelWriter writer = new
DigitalSingleChannelWriter(digitalWriteTask.Stream);
                    writer.WriteSingleSampleSingleLine(true, false);
                }
            }

            if (buffpluse != 0)
            {
                using (Task myTask = new Task())
                {
                    myTask.COChannels.CreatePulseChannelFrequency("Dev1/ctr1",
"PulseTrain3", COPulseFrequencyUnits.Hertz, COPulseIdleState.Low, 0.0, 10000.0, 0.5);

myTask.Timing.ConfigureImplicit(SampleQuantityMode.FiniteSamples,
Convert.ToInt32(Math.Abs(buffpluse) * 262144 / 360));
                    myTask.Start();
```

```
                    do
                    {
                    } while (!myTask.IsDone);
                }
            }
        }
        catch
        {
        }
    }
    #endregion
    // กำหนดตัวแปลที่จะใช้ในการอับเดทการเคลื่อนที่ด้วยมือ
    private double[] _oldangle = new double[3] { 0.0, 0.0, 0.0 };
    private double _FrequencyMH = 800000.0;     // ความถี่ของในการส่งของ MH
200000.0
    private int _IntervalMH = 10;       // เวลาในการวนลูปชอง timer       100
    private double _gapMH = 0.45;        // ค่าช่องว่างของ MH
    // ปุ่มสั่งให้การเคลื่อนที่ด้วยมือ
    private void switch2_StateChanged(object sender, ActionEventArgs e)
    {
        if (switch2.Value)
        {
            for (int i = 0; i < 3; i++)
            {
                _oldangle[i] = numericEditArray1.GetValues()[i];
            }
            timer1.Interval = _IntervalMH; timer1.Enabled = true;   // ให้ Time1 ทำงาน
สำหรับอัพเดทตำแหน่งของ motor1 ทำงาน
            timer2.Interval = _IntervalMH; timer2.Enabled = true;   // ให้ Time2 ทำงาน
สำหรับอัพเดทตำแหน่งของ motor2 ทำงาน
```

```
            timer3.Interval = _IntervalMH; timer3.Enabled = true;   // ให้ Time3 ทำงาน
สำหรับอัพเดทตำแหน่งของ motor3 ทำงาน
            oold = numericEditArray2.GetValues()[2];
            timer5.Enabled = true;
        }
        else
        {
            timer1.Enabled = false;
            timer2.Enabled = false;
            timer3.Enabled = false;
            timer5.Enabled = false;
        }
    }
    // อัพเดทตำแหน่งของ motor1
    private void timer1_Tick(object sender, EventArgs e)
    {
        if (numericEditArray2.GetValues()[2] > Pos[numberObject])
        {
            double buff = _oldangle[0] - numericEditArray1.GetValues()[0];
            if (Math.Abs(buff) > _gapMH)
            {
                try
                {
                    if (buff < 0)
                    {
                        using (Task digitalWriteTask = new Task())
                        {
                            digitalWriteTask.DOChannels.CreateChannel("Dev1/port0/line0",
"DigitalOut_1", ChannelLineGrouping.OneChannelForEachLine);
```

```
                    DigitalSingleChannelWriter writer = new
DigitalSingleChannelWriter(digitalWriteTask.Stream);
                    writer.WriteSingleSampleSingleLine(true, true);
                }
            }
            else if (buff > 0)
            {
                using (Task digitalWriteTask = new Task())
                {
                    digitalWriteTask.DOChannels.CreateChannel("Dev1/port0/line0",
"DigitalOut_1", ChannelLineGrouping.OneChannelForEachLine);
                    DigitalSingleChannelWriter writer = new
DigitalSingleChannelWriter(digitalWriteTask.Stream);
                    writer.WriteSingleSampleSingleLine(true, false);
                }
            }
            if (buff != 0)
            {
                using (Task myTask = new Task())
                {
myTask.COChannels.CreatePulseChannelFrequency("Dev1/ctr0", "PulseTrain1",
COPulseFrequencyUnits.Hertz, COPulseIdleState.Low, 0.0, _FrequencyMH, 0.5);
myTask.Timing.ConfigureImplicit(SampleQuantityMode.FiniteSamples,
Convert.ToInt32(Math.Abs(buff) * 262144 / 360));
                    myTask.Start();
                    do
                    {
                    } while (!myTask.IsDone);
                }
            }
        }
```

```
                catch
                {
                }
                _oldangle[0] = numericEditArray1.GetValues()[0];
            }
        }
    }
    // อัพเดทตำแหน่งของ motor2
    private void timer2_Tick(object sender, EventArgs e)
    {
        if (numericEditArray2.GetValues()[2] > Pos[numberObject])
        {
            double buff = _oldangle[1] - numericEditArray1.GetValues()[1];
            if (Math.Abs(buff) > _gapMH)
            {
                try
                {
                    if (buff < 0)
                    {
                        using (Task digitalWriteTask = new Task())
                        {
                            digitalWriteTask.DOChannels.CreateChannel("Dev1/port0/line2",
"DigitalOut_2", ChannelLineGrouping.OneChannelForEachLine);
                            DigitalSingleChannelWriter writer = new
DigitalSingleChannelWriter(digitalWriteTask.Stream);
                            writer.WriteSingleSampleSingleLine(true, true);
                        }
                    }
                    else if (buff > 0)
                    {
```

```
                    using (Task digitalWriteTask = new Task())
                    {
                        digitalWriteTask.DOChannels.CreateChannel("Dev1/port0/line2",
"DigitalOut_2", ChannelLineGrouping.OneChannelForEachLine);
                        DigitalSingleChannelWriter writer = new
DigitalSingleChannelWriter(digitalWriteTask.Stream);
                        writer.WriteSingleSampleSingleLine(true, false);
                    }
                }

                if (buff != 0)
                {
                    using (Task myTask = new Task())
                    {
myTask.COChannels.CreatePulseChannelFrequency("Dev1/ctr2", "PulseTrain2",
COPulseFrequencyUnits.Hertz, COPulseIdleState.Low, 0.0, _FrequencyMH, 0.5);
 myTask.Timing.ConfigureImplicit(SampleQuantityMode.FiniteSamples,
Convert.ToInt32(Math.Abs(buff) * 262144 / 360));
                        myTask.Start();
                        do
                        {
                        } while (!myTask.IsDone);
                    }
                }
            }
            catch
            {
            }
            _oldangle[1] = numericEditArray1.GetValues()[1];
        }
    }
```

```
        }
        // อัพเดทตำแหน่งของ motor3
        private void timer3_Tick(object sender, EventArgs e)
        {
            if (numericEditArray2.GetValues()[2] > Pos[numberObject])
            {
                double buff = _oldangle[2] - numericEditArray1.GetValues()[2];
                if (Math.Abs(buff) > _gapMH)
                {
                    try
                    {
                        if (buff < 0)
                        {
                            using (Task digitalWriteTask = new Task())
                            {
                                digitalWriteTask.DOChannels.CreateChannel("Dev1/port0/line1",
"DigitalOut_3", ChannelLineGrouping.OneChannelForEachLine);
                                DigitalSingleChannelWriter writer = new
DigitalSingleChannelWriter(digitalWriteTask.Stream);
                                writer.WriteSingleSampleSingleLine(true, true);
                            }
                        }
                        else if (buff > 0)
                        {
                            using (Task digitalWriteTask = new Task())
                            {
                                digitalWriteTask.DOChannels.CreateChannel("Dev1/port0/line1",
"DigitalOut_3", ChannelLineGrouping.OneChannelForEachLine);
                                DigitalSingleChannelWriter writer = new
DigitalSingleChannelWriter(digitalWriteTask.Stream);
```

```
                        writer.WriteSingleSampleSingleLine(true, false);
                    }
                }

                if (buff != 0)
                {
                    using (Task myTask = new Task())
                    {
myTask.COChannels.CreatePulseChannelFrequency("Dev1/ctr1", "PulseTrain3",
COPulseFrequencyUnits.Hertz, COPulseIdleState.Low, 0.0, _FrequencyMH, 0.5);
myTask.Timing.ConfigureImplicit(SampleQuantityMode.FiniteSamples,
Convert.ToInt32(Math.Abs(buff) * 262144 / 360));
                        myTask.Start();
                        do
                        {

                        } while (!myTask.IsDone);
                    }
                }
            }
            catch
            {
            }
            _oldangle[2] = numericEditArray1.GetValues()[2];
        }
    }
}
// ปุ่มสั่งค่าแรงดัน
private void btn_setanalog_Click(object sender, EventArgs e)
{
```

```
            _AnalogOut.AnalogWriter(new double[3] { numericEdit2.Value,
numericEdit2.Value, numericEdit2.Value });
        }
        #endregion
        #region TCP/IP
        public delegate void UpdateRichEditCallback(string text);
        public delegate void UpdateClientListCallback();
        public AsyncCallback pfnWorkerCallBack;
        private Socket m_mainSocket;

        // ArrayList จะใช้ในการติดตามการปฏิบัติงานของซ็อกเก็ตที่ได้รับการออกแบบ
        // การสื่อสารกับลูกค้าที่เชื่อมต่อกัน ทำให้มันเป็น ArrayList ตรง สำหรับความปลอดภัยด้าย
        private System.Collections.ArrayList m_workerSocketList =
                ArrayList.Synchronized(new System.Collections.ArrayList());

        // ตัวแปรต่อไปนี้จะติดตามการสะสม จำนวนรวมของลูกค้าที่เชื่อมต่อได้ตลอดเวลา ตั้งแต่
หลายหัวข้อ
        // สามารถเข้าถึงตัวแปรนี้แก้ไขตัวแปรนี้ควรจะทำ ด้ายในลักษณะที่ปลอดภัย
        private int m_clientCount = 0;
        // หาค่า IP ของเครื่อง
        String GetIP()
        {
            String strHostName = Dns.GetHostName();

            // Find host by name
            IPHostEntry iphostentry = Dns.GetHostByName(strHostName);

            // Grab the first IP addresses
            String IPStr = "";
            foreach (IPAddress ipaddress in iphostentry.AddressList)
```

```
        {
            IPStr = ipaddress.ToString();
            return IPStr;
        }
        return IPStr;
    }
    // เริ่มสร้างการเชื่อมต่อ
    private void btn_Start_Click(object sender, EventArgs e)
    {
        try
        {
            // ตรวจสอบค่า
            if (txt_Port.Text == "")
            {
                MessageBox.Show("Please enter a Port Number");
                return;
            }
            string portStr = txt_Port.Text;
            int port = System.Convert.ToInt32(portStr);
            // สร้างซ็อกเก็ตในการเชื่อมต่อ
            m_mainSocket = new Socket(AddressFamily.InterNetwork,
                SocketType.Stream,
                ProtocolType.Tcp);
            IPEndPoint ipLocal = new IPEndPoint(IPAddress.Any, port);
            // ผูกกับที่อยู่ IP ของเครื่องลูก
            m_mainSocket.Bind(ipLocal);
            // เริ่มการเชื่อมต่อ
            m_mainSocket.Listen(4);
            // สร้างการเรียกกลับสำหรับการเชื่อมต่อของลูกค้าใด ๆ
            m_mainSocket.BeginAccept(new AsyncCallback(OnClientConnect), null);
```

```
            UpdateControls(true);
        }
        catch (SocketException se)
        {
            MessageBox.Show(se.Message);
        }
    }

    // การเรียกกลับซึ่งจะถูกเรียกเมื่อมีการเชื่อมต่อ
    public void OnClientConnect(IAsyncResult asyn)
    {
        try
        {
            // Here we complete/end the BeginAccept() asynchronous call
            // by calling EndAccept() - which returns the reference to
            // a new Socket object
            Socket workerSocket = m_mainSocket.EndAccept(asyn);
            // Now increment the client count for this client
            // in a thread safe manner
            Interlocked.Increment(ref m_clientCount);
            // Add the workerSocket reference to our ArrayList
            m_workerSocketList.Add(workerSocket);
            // Send a welcome message to client
            string msg = "Welcome client " + m_clientCount + "\n";
            SendMsgToClient(msg, m_clientCount);

            // Update the list box showing the list of clients (thread safe call)
            UpdateClientListControl();

            // Let the worker Socket do the further processing for the
```

```
            // just connected client
            WaitForData(workerSocket, m_clientCount);

            // Since the main Socket is now free, it can go back and wait for
            // other clients who are attempting to connect
            m_mainSocket.BeginAccept(new AsyncCallback(OnClientConnect), null);
        }
        catch (ObjectDisposedException)
        {
            System.Diagnostics.Debugger.Log(0, "1", "\n OnClientConnection: Socket
has been closed\n");
        }
        catch (SocketException se)
        {
            MessageBox.Show(se.Message);
        }
    }
    // ส่งข้อความไปยังลูกค้า
    void SendMsgToClient(string msg, int clientNumber)
    {
        // Convert the reply to byte array
        byte[] byData = System.Text.Encoding.ASCII.GetBytes(msg);

        Socket workerSocket = (Socket)m_workerSocketList[clientNumber - 1];
        workerSocket.Send(byData);
    }
    private void UpdateClientListControl()
    {
        if (InvokeRequired) // Is this called from a thread other than the one created
        // the control
        {
```

```
                // We cannot update the GUI on this thread.
                // All GUI controls are to be updated by the main (GUI) thread.
                // Hence we will use the invoke method on the control which will
                // be called when the Main thread is free
                // Do UI update on UI thread
                lb_ClientList.BeginInvoke(new UpdateClientListCallback(UpdateClientList),
null);
            }
            else
            {
                // This is the main thread which created this control, hence update it
                // directly
                UpdateClientList();
            }
        }
        // อัพเดตกล่องรายการแสดงรายการของลูกค้า
        void UpdateClientList()
        {
            lb_ClientList.Items.Clear();
            for (int i = 0; i < m_workerSocketList.Count; i++)
            {
                string clientKey = Convert.ToString(i + 1);
                Socket workerSocket = (Socket)m_workerSocketList[i];
                if (workerSocket != null)
                {
                    if (workerSocket.Connected)
                    {
                        lb_ClientList.Items.Add(clientKey);
                    }
                }
```

```
        }
    }

    // เริ่มรอข้อมูลจากลูกค้า
    public void WaitForData(System.Net.Sockets.Socket soc, int clientNumber)
    {
        try
        {
            if (pfnWorkerCallBack == null)
            {
                // Specify the call back function which is to be
                // invoked when there is any write activity by the
                // connected client
                pfnWorkerCallBack = new AsyncCallback(OnDataReceived);
            }
            SocketPacket theSocPkt = new SocketPacket(soc, clientNumber);

            soc.BeginReceive(theSocPkt.dataBuffer, 0,
                theSocPkt.dataBuffer.Length,
                SocketFlags.None,
                pfnWorkerCallBack,
                theSocPkt);
        }
        catch (SocketException se)
        {
            MessageBox.Show(se.Message);
        }
    }

    public class SocketPacket
    {
```

```
        // Constructor which takes a Socket and a client number
        public SocketPacket(System.Net.Sockets.Socket socket, int clientNumber)
        {
            m_currentSocket = socket;
            m_clientNumber = clientNumber;
        }
        public System.Net.Sockets.Socket m_currentSocket;
        public int m_clientNumber;
        // Buffer to store the data sent by the client
        public byte[] dataBuffer = new byte[1024];
    }

    // This the call back function which will be invoked when the socket
    // detects any client writing of data on the stream
    public void OnDataReceived(IAsyncResult asyn)
    {
        SocketPacket socketData = (SocketPacket)asyn.AsyncState;
        try
        {
            // Complete the BeginReceive() asynchronous call by EndReceive()
method
            // which will return the number of characters written to the stream
            // by the client
            int iRx = socketData.m_currentSocket.EndReceive(asyn);
            char[] chars = new char[iRx + 1];
            // Extract the characters as a buffer
            System.Text.Decoder d = System.Text.Encoding.UTF8.GetDecoder();
            int charLen = d.GetChars(socketData.dataBuffer,
                0, iRx, chars, 0);
```

```
            System.String szData = new System.String(chars);
            string msg = "" + socketData.m_clientNumber + ":";
            //AppendToRichEditControl(msg + szData);

            if (socketData.m_clientNumber == 1)
            {
                Spilttext(szData);
            }
            //// Send back the reply to the client
            //string replyMsg = "Server Reply:" + szData.ToUpper();
            //// Convert the reply to byte array
            //byte[] byData = System.Text.Encoding.ASCII.GetBytes(replyMsg);

            //Socket workerSocket = (Socket)socketData.m_currentSocket;
            //workerSocket.Send(byData);

            // Continue the waiting for data on the Socket
            WaitForData(socketData.m_currentSocket, socketData.m_clientNumber);

        }
        catch (ObjectDisposedException)
        {
            System.Diagnostics.Debugger.Log(0, "1", "\nOnDataReceived: Socket has
been closed\n");
        }
        catch (SocketException se)
        {
            if (se.ErrorCode == 10054) // Error code for Connection reset by peer
            {
                string msg = "Client " + socketData.m_clientNumber + " Disconnected"
+ "\n";
```

```
                AppendToRichEditControl(msg);

                // Remove the reference to the worker socket of the closed client
                // so that this object will get garbage collected
                m_workerSocketList[socketData.m_clientNumber - 1] = null;
                UpdateClientListControl();
            }
            else
            {
                MessageBox.Show(se.Message);
            }
        }
    }

    // This method could be called by either the main thread or any of the
    // worker threads
    private void AppendToRichEditControl(string msg)
    {
        // Check to see if this method is called from a thread
        // other than the one created the control
        if (InvokeRequired)
        {
            // We cannot update the GUI on this thread.
            // All GUI controls are to be updated by the main (GUI) thread.
            // Hence we will use the invoke method on the control which will
            // be called when the Main thread is free
            // Do UI update on UI thread
            object[] pList = { msg };
            richTextBoxReceivedMsg.BeginInvoke(new
UpdateRichEditCallback(OnUpdateRichEdit), pList);
```

```
        }
        else
        {
            // This is the main thread which created this control, hence update it
            // directly
            OnUpdateRichEdit(msg);
        }
    }

    // This UpdateRichEdit will be run back on the UI thread
    // (using System.EventHandler signature
    // so we don't need to define a new
    // delegate type here)
    private void OnUpdateRichEdit(string msg)
    {
        richTextBoxReceivedMsg.AppendText(msg + "\n");
    }

    // ปุ่มหยุดการเชื่อมต่อ
    private void btn_Stop_Click(object sender, EventArgs e)
    {
        CloseSockets();
        UpdateControls(false);
    }

    // หยุดการเชื่อมต่อ
    void CloseSockets()
    {
        if (m_mainSocket != null)
        {
            m_mainSocket.Close();
```

```
    }
    Socket workerSocket = null;
    for (int i = 0; i < m_workerSocketList.Count; i++)
    {
        workerSocket = (Socket)m_workerSocketList[i];
        if (workerSocket != null)
        {
            workerSocket.Close();
            workerSocket = null;
        }
    }
}

// เช็คปุ่ม
private void UpdateControls(bool listening)
{
    btn_Start.Enabled = !listening;
    btn_Stop.Enabled = listening;
}

// ปุ่มส่งข้อความ
private void btn_SendMsg_Click(object sender, EventArgs e)
{
    try
    {
        string msg = richTextBoxSendMsg.Text;
        //msg = "Server Msg: " + msg + "\n";
        byte[] byData = System.Text.Encoding.ASCII.GetBytes(msg);
        Socket workerSocket = null;
        for (int i = 0; i < m_workerSocketList.Count; i++)
```

```
            {
                workerSocket = (Socket)m_workerSocketList[i];
                if (workerSocket != null)
                {
                    if (workerSocket.Connected)
                    {
                        workerSocket.Send(byData);
                    }
                }
            }
        }
        catch (SocketException se)
        {
            MessageBox.Show(se.Message);
        }
    }

    // ปุ่มสุ่มตัวเลขของข้อมูล
    private void btn_Random_Click(object sender, EventArgs e)
    {
        Random num1 = new Random();
        richTextBoxSendMsg.Text = "[" + num1.Next(-100, 100).ToString() + "," +
num1.Next(-100, 100).ToString() + "," + num1.Next(-100, 100).ToString() + "]";
    }

    // ตัดตัวหนังสือที่ส่งเข้ามา
    private void Spilttext(string msg)
    {
        //Invoke(new EventHandler(delegate
        //    {
        //        listBox1.Items.Clear();
```

```
            //        List<string> _items = new List<string>();
            //        string configText = msg;
            //        string[] configTexts = configText.Split(' ', '{', '}');
            //        foreach (string item in configTexts)
            //        {
            //            if ("" != item)
            //            {
            //                _AnalogOut.AnalogWriter(new double[3] { numericEdit2.Value,
numericEdit2.Value, numericEdit2.Value });
            //            }
            //        }
            //    }));
            if (msg == "A\r\n\0")
            {
                numberObject = 0;
            }
            else if (msg == "B\r\n\0")
            {
                numberObject = 1;
            }
            else
            {
                numberObject = 2;
            }
            switch (numberObject)
            {
                case 0: textBox1.Invoke((MethodInvoker)delegate { textBox1.Text =
"Object Green"; }); break;
```

```
                case 1: textBox1.Invoke((MethodInvoker)delegate { textBox1.Text =
"Object Red"; }); break;
                case 2: textBox1.Invoke((MethodInvoker)delegate { textBox1.Text = "NO
Object"; }); break;
                default:
                    break;
            }
        }
        #endregion
        #region Communication
        private double[] _oldposition = new double[3] { 0.0, 0.0, -383.7642 };
        // ปุ่มส่งข้อมูลไปยัง ABB Robot
        private void switch3_StateChanged(object sender, ActionEventArgs e)
        {
            if (switch2.Value)
            {
                for (int i = 0; i < 3; i++)
                {
                    _oldposition[i] = numericEditArray2.GetValues()[i];
                }
                timer4.Enabled = true;
            }
            else
            {
                timer4.Enabled = false;
            }
        }

        // timer ที่ใช้ในการส่งข้อมูลไปยัง ABB Rbobt
        private void timer4_Tick(object sender, EventArgs e)
        {
```

```
            try
            {
                string msg = "[" + numericEditArray2.GetValues()[1].ToString("0.0") + "," +
numericEditArray2.GetValues()[0].ToString("0.0") + "," + (-
(numericEditArray2.GetValues()[2] + 383.7642)).ToString("0.0") + "]";
                richTextBoxSendMsg.Text = msg;
                byte[] byData = System.Text.Encoding.ASCII.GetBytes(msg);
                Socket workerSocket = null;
                for (int i = 0; i < m_workerSocketList.Count; i++)
                {
                    workerSocket = (Socket)m_workerSocketList[i];
                    if (workerSocket != null)
                    {
                        if (workerSocket.Connected)
                        {
                            workerSocket.Send(byData);
                        }
                    }
                }
            }
            catch (SocketException se)
            {
                MessageBox.Show(se.Message);
            }
        }

        #endregion
        double oold = 0;
        private void timer5_Tick(object sender, EventArgs e)
        {
```

```
            if (numberObject == 2)
            {
                if (numericEdit2.Value == 1.6)
                {
                    return;
                }
                numericEdit2.Value = 1.6;
                _AnalogOut.AnalogWriter(new double[3] { numericEdit2.Value,
numericEdit2.Value, numericEdit2.Value });
                return;
            }
            if (Pos[numberObject] >= numericEditArray2.GetValues()[2])
            {
                if (oold < numericEditArray2.GetValues()[2])
                {
                    if ((oold - numericEditArray2.GetValues()[2]) > 0.1)
                    {
                        numericEdit2.Value = 1.6;
                        _AnalogOut.AnalogWriter(new double[3] { numericEdit2.Value,
numericEdit2.Value, numericEdit2.Value });
                        oold = numericEditArray2.GetValues()[2];
                    }
                }
                else
                {
                    numericEdit2.Value =
_kinematics.map(numericEditArray2.GetValues()[2], setmaxT[numberObject],
Pos[numberObject], 7, 1.6);
                    if (numericEdit2.Value > 7) numericEdit2.Value = 7;
                    _AnalogOut.AnalogWriter(new double[3] { numericEdit2.Value,
numericEdit2.Value, numericEdit2.Value });
```

```
            }
        }
        else
        {
            if (numericEdit2.Value == 1.6)
            {
                return;
            }
            numericEdit2.Value = 1.6;
            _AnalogOut.AnalogWriter(new double[3] { numericEdit2.Value,
numericEdit2.Value, numericEdit2.Value });
        }
    }
    // กำหนดตัวแปลของตำแหน่งหุ่นยนต์
    double[] setmaxT = new double[2] { -540, -464.0 };
    double[] Pos = new double[3] { -479, -455, -600.0 };
    int numberObject = 2;
    private void button2_Click(object sender, EventArgs e)
    {
        Pos[0] = numericEdit3.Value;
        Pos[1] = numericEdit4.Value;
    }
}

// struct ที่ใช้เก็บค่าขององศามอเตอร์
public struct MotorAngle
{
    private double _AngleM1;
    private double _AngleM2;
    private double _AngleM3;
```

```
        public double AngleM1
        {
            get { return _AngleM1; }
            set { _AngleM1 = value; }
        }
        public double AngleM2
        {
            get { return _AngleM2; }
            set { _AngleM2 = value; }
        }
        public double AngleM3
        {
            get { return _AngleM3; }
            set { _AngleM3 = value; }
        }
        public double[] Array
        {
            get { return new double[] { _AngleM1, _AngleM2, _AngleM3 }; }
            set
            {
                if (value.Length == 3)
                {
                    _AngleM1 = value[0];
                    _AngleM2 = value[1];
                    _AngleM3 = value[2];
                }
                else
                {
                    throw new ArgumentException("Array must contain exactly three
components , (M1 M2 M3)");
                }
```

```
        }
    }
    public double this[int index]
    {
        get
        {
            switch (index)
            {
                case 0: { return AngleM1; }
                case 1: { return AngleM2; }
                case 2: { return AngleM3; }
                default: throw new ArgumentException("Array must contain exactly
three components , (M1 M2 M3)", "index");
            }
        }
        set
        {
            switch (index)
            {
                case 0: { AngleM1 = value; break; }
                case 1: { AngleM2 = value; break; }
                case 2: { AngleM3 = value; break; }
                default: throw new ArgumentException("Array must contain exactly
three components , (M1 M2 M3)", "index");
            }
        }
    }

    public MotorAngle(double m1, double m2, double m3)
    {
```

```
            _AngleM1 = 0.0;
            _AngleM2 = 0.0;
            _AngleM3 = 0.0;

            AngleM1 = m1;
            AngleM2 = m2;
            AngleM3 = m3;
        }
        public MotorAngle(double[] mall)
        {
            _AngleM1 = 0.0;
            _AngleM2 = 0.0;
            _AngleM3 = 0.0;

            Array = mall;
        }
        public MotorAngle(MotorAngle v1)
        {
            _AngleM1 = 0.0;
            _AngleM2 = 0.0;
            _AngleM3 = 0.0;

            AngleM1 = v1.AngleM1;
            AngleM2 = v1.AngleM2;
            AngleM3 = v1.AngleM3;
        }
    }
    // มีการเรียกกลับของค่าองศามอเตอร์
    public delegate void getEncoder(MotorAngle value1);
    // คลาสอ่านองศามอเตอร์
    class ReadEncoder
```

```
    {
        private getEncoder writeEncoder;
        private Thread runThread;
        private MotorAngle angle;

        private const int _jointCount = 3;
        private Task[] _ChannelsCI = new Task[_jointCount];
        private CounterReader[] _CounterReader = new CounterReader[_jointCount];
        private string[] _counter = new string[_jointCount] { "Dev2/ctr1", "Dev1/ctr3",
"Dev2/ctr0" };
        private string[] _nameToAssignChannelEncoder = new string[_jointCount] {
"EncoderChannel 1", "EncoderChannel 2", "EncoderChannel3" };
        private double[] _zIndexValue = new double[_jointCount] { 46 - 4.7802, 33.3 -
4.7802, 10.5 - 4.7802 };        //46, 10.5, 33.3
        public bool PulseZ { get; private set; }
        public int timeSleep { get; private set; }
        public bool IsRun { get; private set; }
        public ReadEncoder(getEncoder outFunc, int timeSleep, bool PulseZ)
        {
            writeEncoder = outFunc;
            this.timeSleep = timeSleep;
            this.PulseZ = PulseZ;
        }
        ~ReadEncoder()
        {
            Disable();
            writeEncoder = null;
        }
        public void Enable()
        {
```

```
            if (runThread != null) return;
            for (int i = 0; i < _jointCount; i++)
            {
                _ChannelsCI[i] = new Task();
                _ChannelsCI[i].CIChannels.CreateAngularEncoderChannel(_counter[i],
_nameToAssignChannelEncoder[i], CIEncoderDecodingType.X4, PulseZ,
_zIndexValue[i], CIEncoderZIndexPhase.AHighBHigh, 25000, 999.9999,
CIAngularEncoderUnits.Degrees);
                _CounterReader[i] = new CounterReader(_ChannelsCI[i].Stream);
                _ChannelsCI[i].Start();
            }
            runThread = new Thread(new ThreadStart(Run));
            runThread.IsBackground = true;
            runThread.Name = "Delta Run";
            runThread.Start();
            IsRun = true;
        }
        public void Disable()
        {
            if (runThread != null) { runThread.Abort(); runThread = null; }
            for (int i = 0; i < _jointCount; i++)
            {
                if (_ChannelsCI[i] != null)
                {
                    //_ChannelsCI[i].Stop();
                    _ChannelsCI[i].Dispose();
                }
            }
            IsRun = false;
        }
        private void Run()
```

```
        {
            while (true)
            {
                try
                {
                    Thread.Sleep(timeSleep);
                    for (int i = 0; i < _jointCount; i++)
                    {
                        angle[i] = _CounterReader[i].ReadSingleSampleDouble();
                    }
                    writeEncoder(angle);
                }
                catch
                {
                    writeEncoder = null;
                }
            }
        }
    }

    // คลาสสั่งดิจิตอล
    class DigitalOut
    {
        private delegate void dlgDigital(UInt32 Digital);
        public DigitalOut(UInt32 valu)
        {
            using (Task digitalWriteTask = new Task())
            {
                digitalWriteTask.DOChannels.CreateChannel("Dev1/port0", "digital output
port0", ChannelLineGrouping.OneChannelForAllLines);
```

```
                DigitalSingleChannelWriter writer = new
DigitalSingleChannelWriter(digitalWriteTask.Stream);
                writer.WriteSingleSamplePort(true, valu);
            }
        }
        public void DigitalWrite(UInt32 valu)
        {
            dlgDigital dlgdigital = new dlgDigital(Write);
            dlgdigital.BeginInvoke(valu, null, null);
        }
        private void Write(UInt32 valu)
        {
            using (Task digitalWriteTask = new Task())
            {
                digitalWriteTask.DOChannels.CreateChannel("Dev1/port0", "digital output
port0", ChannelLineGrouping.OneChannelForAllLines);
                DigitalSingleChannelWriter writer = new
DigitalSingleChannelWriter(digitalWriteTask.Stream);
                writer.WriteSingleSamplePort(true, valu);
            }
        }
    }
    // คลาสจ่ายความถี่ให้มอเตอร์
    class Pulse
    {
        public delegate void _dlgPulse(double angle, double Frequency);

        public void PulseGeneration(double angle1, double Frequency1, double
angle2, double Frequency2, double angle3, double Frequency3)
        {
            _dlgPulse dlg1 = new _dlgPulse(Gotoposition1);
```

```
            dlg1.BeginInvoke(angle1, Frequency1, null, null);
            _dlgPulse dlg2 = new _dlgPulse(Gotoposition2);
            dlg2.BeginInvoke(angle2, Frequency2, null, null);
            _dlgPulse dlg3 = new _dlgPulse(Gotoposition3);
            dlg3.BeginInvoke(angle3, Frequency3, null, null);
        }
        private void Gotoposition1(double anglenew, double frequency)
        {
            try
            {
                if (anglenew < 0)
                {
                    using (Task digitalWriteTask = new Task())
                    {
                        digitalWriteTask.DOChannels.CreateChannel("Dev1/port0/line0",
"DigitalOut_1", ChannelLineGrouping.OneChannelForEachLine);
                        DigitalSingleChannelWriter writer = new
DigitalSingleChannelWriter(digitalWriteTask.Stream);
                        writer.WriteSingleSampleSingleLine(true, true);
                    }
                }
                else if (anglenew > 0)
                {
                    using (Task digitalWriteTask = new Task())
                    {
                        digitalWriteTask.DOChannels.CreateChannel("Dev1/port0/line0",
"DigitalOut_1", ChannelLineGrouping.OneChannelForEachLine);
                        DigitalSingleChannelWriter writer = new
DigitalSingleChannelWriter(digitalWriteTask.Stream);
                        writer.WriteSingleSampleSingleLine(true, false);
```

```
                }
            }
            if (anglenew != 0)
            {
                using (Task myTask = new Task())
                {
                    myTask.COChannels.CreatePulseChannelFrequency("Dev1/ctr0",
"PulseTrain1", COPulseFrequencyUnits.Hertz, COPulseIdleState.Low, 0.0, frequency,
0.5);

myTask.Timing.ConfigureImplicit(SampleQuantityMode.FiniteSamples,
Convert.ToInt32(Math.Abs(anglenew) * 262144 / 360));
                    myTask.Start();
                    do
                    {
                    } while (!myTask.IsDone);
                }
            }
        }
        catch
        {
        }
    }
    private void Gotoposition2(double anglenew, double frequency)
    {
        try
        {
            if (anglenew < 0)
            {
                using (Task digitalWriteTask = new Task())
                {
```

```
                    digitalWriteTask.DOChannels.CreateChannel("Dev1/port0/line2",
"DigitalOut_2", ChannelLineGrouping.OneChannelForEachLine);
                    DigitalSingleChannelWriter writer = new
DigitalSingleChannelWriter(digitalWriteTask.Stream);
                    writer.WriteSingleSampleSingleLine(true, true);
                }
            }
            else if (anglenew > 0)
            {
                using (Task digitalWriteTask = new Task())
                {
                    digitalWriteTask.DOChannels.CreateChannel("Dev1/port0/line2",
"DigitalOut_2", ChannelLineGrouping.OneChannelForEachLine);
                    DigitalSingleChannelWriter writer = new
DigitalSingleChannelWriter(digitalWriteTask.Stream);
                    writer.WriteSingleSampleSingleLine(true, false);
                }
            }

            if (anglenew != 0)
            {
                using (Task myTask = new Task())
                {
                    myTask.COChannels.CreatePulseChannelFrequency("Dev1/ctr2",
"PulseTrain2", COPulseFrequencyUnits.Hertz, COPulseIdleState.Low, 0.0, frequency,
0.5);
myTask.Timing.ConfigureImplicit(SampleQuantityMode.FiniteSamples,
Convert.ToInt32(Math.Abs(anglenew) * 262144 / 360));
                    myTask.Start();
                    do
```

```
                    {
                    } while (!myTask.IsDone);
                }
            }
        }
        catch
        {
        }
    }

    private void Gotoposition3(double anglenew, double frequency)
    {
        try
        {
            if (anglenew < 0)
            {
                using (Task digitalWriteTask = new Task())
                {
                    digitalWriteTask.DOChannels.CreateChannel("Dev1/port0/line1",
"DigitalOut_3", ChannelLineGrouping.OneChannelForEachLine);
                    DigitalSingleChannelWriter writer = new
DigitalSingleChannelWriter(digitalWriteTask.Stream);
                    writer.WriteSingleSampleSingleLine(true, true);
                }
            }
            else if (anglenew > 0)
            {
                using (Task digitalWriteTask = new Task())
                {
                    digitalWriteTask.DOChannels.CreateChannel("Dev1/port0/line1",
"DigitalOut_3", ChannelLineGrouping.OneChannelForEachLine);
```

```
                    DigitalSingleChannelWriter writer = new
DigitalSingleChannelWriter(digitalWriteTask.Stream);
                    writer.WriteSingleSampleSingleLine(true, false);
                }
            }
            if (anglenew != 0)
            {
                using (Task myTask = new Task())
                {

                    myTask.COChannels.CreatePulseChannelFrequency("Dev1/ctr1",
"PulseTrain3", COPulseFrequencyUnits.Hertz, COPulseIdleState.Low, 0.0, frequency,
0.5);

                   myTask.Timing.ConfigureImplicit(SampleQuantityMode.FiniteSamples,
Convert.ToInt32(Math.Abs(anglenew) * 262144 / 360));
                    myTask.Start();
                    do
                    {

                    }
while (!myTask.IsDone);
                }
            }
        }

        catch
        {

        }
```

```
        }

    }
    // คลาสสั่งแรงดัน
    class AnalogOut
    {
        private const int _jointCount = 3;
        private string[] _physicalChannelName = new string[_jointCount] { "Dev1/ao0",
"Dev1/ao2", "Dev1/ao1" };
        private string[] _nameToAssignChannelAnalog = new string[_jointCount] {
"Voltage Out 1", "Voltage Out 2", "Voltage Out 3" };
        private delegate void dlgAnalog(double[] Analog);

        public AnalogOut(double[] Analog)
        {
            for (int i = 0; i < _jointCount; i++)
            {
                using (Task ChannelsAO = new Task())
                {
ChannelsAO.AOChannels.CreateVoltageChannel(_physicalChannelName[i],
_nameToAssignChannelAnalog[i], 0, 8.0, AOVoltageUnits.Volts);
                    AnalogSingleChannelWriter writer = new
AnalogSingleChannelWriter(ChannelsAO.Stream);
                    writer.WriteSingleSample(true, Analog[i]);
                }
            }
        }
        public void AnalogWriter(double[] Analog)
        {
            dlgAnalog dlganalod = new dlgAnalog(Writer);
            dlganalod.BeginInvoke(Analog, null, null);
```

```
        }
        private void Writer(double[] Analog)
        {
            for (int i = 0; i < _jointCount; i++)
            {
                using (Task ChannelsAO = new Task())
                {
ChannelsAO.AOChannels.CreateVoltageChannel(_physicalChannelName[i],
_nameToAssignChannelAnalog[i], 0, 8.0, AOVoltageUnits.Volts);

                    AnalogSingleChannelWriter writer = new
AnalogSingleChannelWriter(ChannelsAO.Stream);

                    writer.WriteSingleSample(true, Analog[i]);
                }
            }
        }
    }

// คลาสในการคำนวน kinematics ของหุ่นยนต์
    class kinematics
    {
 // สร้างตัวแปรที่เป็นค่าคงที่
        private double sqrt3 = Math.Sqrt(3.0);
        private double pi = Math.PI;     // PI
        private double sin120 = Math.Sqrt(3.0) / 2.0;
        private double cos120 = -0.5;
        private double tan60 = Math.Sqrt(3.0);
        private double sin30 = 0.5;
        private double tan30 = 1.0 / Math.Sqrt(3.0);
```

```
// ตัวแปรที่ใช้ในการคำนวนได้มาจากการออกแบบหุ่นยนต์
public double f { get; set; }       // ความยาวด้านหนึ่งของสามเหลี่ยมบน
public double e { get; set; }       // ความยาวด้านนึ่งของสามเหลี่ยมล่าง
public double rf { get; set; }      // ความยาวของแขนบน
public double re { get; set; }      // ความยาวของแขนล่าง
// การทำงานในส่วนของคอนสทรักเตอร์ ตอนสร้างอินสแทนซ์
public kinematics(double f, double e, double rf, double re)
{
    this.f = f;
    this.e = e;
    this.rf = rf;
    this.re = re;
}
// คำนวณไปข้างหน้า
public double[] calcForward(double theta1, double theta2, double theta3)
{

    double _x0 = 0.0;
    double _y0 = 0.0;
    double _z0 = 0.0;
    double t = (f - e) * tan30 / 2.0;
    double dtr = pi / 180.0;
    theta1 *= dtr;
    theta2 *= dtr;
    theta3 *= dtr;
    double y1 = -(t + rf * Math.Cos(theta1));
    double z1 = -rf * Math.Sin(theta1);
    double y2 = (t + rf * Math.Cos(theta2)) * sin30;
    double x2 = y2 * tan60;
    double z2 = -rf * Math.Sin(theta2);
    double y3 = (t + rf * Math.Cos(theta3)) * sin30;
```

```
            double x3 = -y3 * tan60;
            double z3 = -rf * Math.Sin(theta3);
            double dnm = (y2 - y1) * x3 - (y3 - y1) * x2;
            double w1 = y1 * y1 + z1 * z1;
            double w2 = x2 * x2 + y2 * y2 + z2 * z2;
            double w3 = x3 * x3 + y3 * y3 + z3 * z3;
            // x = (a1*z + b1)/dnm
            double a1 = (z2 - z1) * (y3 - y1) - (z3 - z1) * (y2 - y1);
            double b1 = -((w2 - w1) * (y3 - y1) - (w3 - w1) * (y2 - y1)) / 2.0;
            // y = (a2*z + b2)/dnm;
            double a2 = -(z2 - z1) * x3 + (z3 - z1) * x2;
            double b2 = ((w2 - w1) * x3 - (w3 - w1) * x2) / 2.0;
            // a*z^2 + b*z + c = 0
            double a = a1 * a1 + a2 * a2 + dnm * dnm;
            double b = 2.0 * (a1 * b1 + a2 * (b2 - y1 * dnm) - z1 * dnm * dnm);
            double c = (b2 - y1 * dnm) * (b2 - y1 * dnm) + b1 * b1 + dnm * dnm * (z1 *
z1 - re * re);          // จำแนกข้อมูล
            double d = b * b - 4.0 * a * c;
            if (d < 0.0) return new double[4] { -1, 0, 0, 0 }; // ไม่สามารถหาค่าได้
            _z0 = -0.5 * (b + Math.Sqrt(d)) / a;
            _x0 = (a1 * _z0 + b1) / dnm;
            _y0 = (a2 * _z0 + b2) / dnm;
            return new double[4] { 0, _x0, _y0, _z0 };
        }
        // คำนวนกลับ
        public double[] calcInverse(double x0, double y0, double z0)
        {
            double _Theta1 = 0;
            double _Theta2 = 0;
            double _Theta3 = 0;
```

```
            double[] status = calcAngleYZ(x0, y0, z0, _Theta1);
            if (status[0] == 0)
            {
                _Theta1 = status[1];
                status = calcAngleYZ(x0 * cos120 + y0 * sin120, y0 * cos120 - x0 * sin120,
z0, _Theta2);  // หมุนไป +120 deg
            }
            if (status[0] == 0)
            {
                _Theta2 = status[1];
                status = calcAngleYZ(x0 * cos120 - y0 * sin120, y0 * cos120 + x0 * sin120,
z0, _Theta3);  // หมุนไป -120 deg
            }
            _Theta3 = status[1];
            return new double[4] { 0, _Theta1, _Theta2, _Theta3 };
        }
        private double[] calcAngleYZ(double x0, double y0, double z0, double theta)
        {
            double y1 = -0.5 * 0.57735 * f;  // f/2 * tg 30
            y0 -= 0.5 * 0.57735 * e;  // shift center to edge
            // z = a + b*y
            double a = (x0 * x0 + y0 * y0 + z0 * z0 + rf * rf - re * re - y1 * y1) / (2.0 * z0);
            double b = (y1 - y0) / z0;
            // จำแนกข้อมูล
            double d = -(a + b * y1) * (a + b * y1) + rf * (b * b * rf + rf);
            if (d < 0) return new double[2] { -1, 0 }; // ไม่สามารถหาค่าได้
            double yj = (y1 - a * b - Math.Sqrt(d)) / (b * b + 1);
            double zj = a + b * yj;
            theta = Math.Atan(-zj / (y1 - yj)) * 180.0 / pi + ((yj > y1) ? 180.0 : 0.0);
            return new double[2] { 0, theta };
        }
```

```
        // หาอาศาของตัวโรเอ็น
        public double[] calctheta_rodend(double x0, double y0)
        {
            double Cy1 = y0 * -Math.Sin(DegreeToRadian(180)) + x0 *
Math.Cos(DegreeToRadian(180));
            double Cy2 = y0 * -Math.Sin(DegreeToRadian(60)) + x0 *
Math.Cos(DegreeToRadian(60));
            double Cy3 = y0 * -Math.Sin(DegreeToRadian(-60)) + x0 *
Math.Cos(DegreeToRadian(-60));
            double rodend1 = RadianToDegree(Math.Acos(Cy1 / re));
            double rodend2 = RadianToDegree(Math.Acos(Cy2 / re));
            double rodend3 = RadianToDegree(Math.Acos(Cy3 / re));

            if (rodend1 >= 90)
            {
                rodend1 = 90 - rodend1;
            }
            else
            {
                rodend1 = 90 - rodend1;
            }
            if (rodend2 >= 90)
            {
                rodend2 = 90 - rodend2;
            }
            else
            {
                rodend2 = 90 - rodend2;
            }
```

```
            if (rodend3 >= 90)
            {
                rodend3 = 90 - rodend3;
            }
            else
            {
                rodend3 = 90 - rodend3;
            }
            return new double[4] { 0, rodend1, rodend2, rodend3 };
        }
        // แปลง DegreeToRadian
        private double DegreeToRadian(double angle)
        {
            return Math.PI * angle / 180.0;
        }
        // แปลง RadianToDegree
        private double RadianToDegree(double angle)
        {
            return angle * (180.0 / Math.PI);
        }
        // หาค่าช่วงของ input-output
        public double map(double x, double in_min, double in_max, double out_min,
double out_max)
        {
            return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
        }
    }
}
```