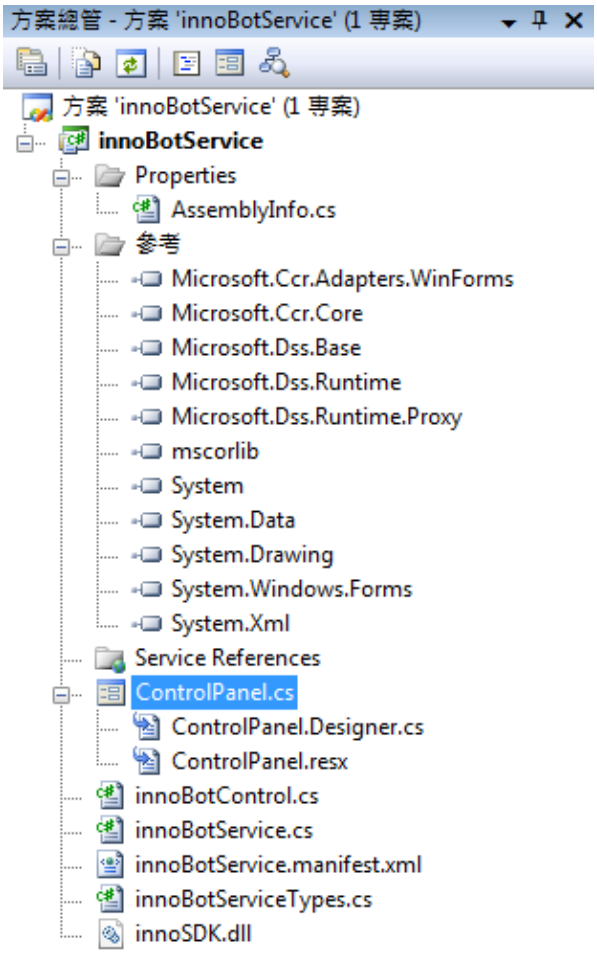
innoBotService

innoBotService Project Files



innoBotControl.cs

```
using System;
using System.Collections.Generic;
//using System.Linq;
using System.Text;
using System.Runtime.InteropServices;

namespace innoBotService
{
    class innoBotControl
    {
```

```
15          [DllImport("innoSDK.dll", SetLastError = true, CharSet = CharSet.Unicode)]
16          internal static extern bool inno_Connect();
17
18          [DllImport("innoSDK.dll", SetLastError = true, CharSet = CharSet.Unicode)]
19          internal static extern void inno_Disconnect();
20
21          [DllImport("innoSDK.dll", SetLastError = true, CharSet = CharSet.Unicode)]
22          internal static extern bool inno_IsConnect();
23
24          [DllImport("innoSDK.dll", SetLastError = true, CharSet = CharSet.Unicode)]
25          internal static extern System.UInt32 inno_ResetTarget();
26
27          //       DWORD PASCAL EXPORT Sonar_SetPosition(WORD a_InPos);
28          [DllImport("innoSDK.dll", SetLastError = true, CharSet = CharSet.Unicode)]
29          internal static extern System.UInt32 Sonar_SetPosition(System.UInt16 a_InPos);
30
31          //       DWORD PASCAL EXPORT inno_DebugInLong(int a_inLongData);
32          [DllImport("innoSDK.dll", SetLastError = true, CharSet = CharSet.Unicode)]
33          internal static extern System.UInt32 inno_DebugInLong(System.Int32 a_InSpeed);
34
35          //       DWORD PASCAL EXPORT inno_DebugInFloat(float a_infData);
36          [DllImport("innoSDK.dll", SetLastError = true, CharSet = CharSet.Unicode)]
37          internal static extern System.UInt32 inno_DebugInFloat(System.Single a_infData);
38
39          //       DWORD PASCAL EXPORT inno_DebugInInteger(short a_inshortData);
40          [DllImport("innoSDK.dll", SetLastError = true, CharSet = CharSet.Unicode)]
41          internal static extern System.UInt32 inno_DebugInInteger(System.UInt16 a_inshortData);
42
43          //       DWORD PASCAL EXPORT inno_DebugInSByte(char a_insbyteData);
44          [DllImport("innoSDK.dll", SetLastError = true, CharSet = CharSet.Unicode)]
45          internal static extern System.UInt32 inno_DebugInSByte(System.Char a_insbyteData);
46
47          //       DWORD PASCAL EXPORT inno_DebugInByte(BYTE a_inByteData);
48          [DllImport("innoSDK.dll", SetLastError = true, CharSet = CharSet.Unicode)]
49          internal static extern System.UInt32 inno_DebugInByte(System.Byte a_inByteData);
50
51          //       DWORD PASCAL EXPORT inno_DebugInDWORD(unsigned int a_inDWORDData);
52          [DllImport("innoSDK.dll", SetLastError = true, CharSet = CharSet.Unicode)]
53          internal static extern System.UInt32 inno_DebugInDWORD(System.UInt32 a_inDWORDData);
54
```

```
        //       DWORD PASCAL EXPORT inno_DebugInWORD(unsigned short a_inWORDData);
        [DllImport("innoSDK.dll", SetLastError = true, CharSet = CharSet.Unicode)]
        internal static extern System.UInt32 inno_DebugInWORD(System.UInt16 a_inWORDData);

        //       DWORD PASCAL EXPORT inno_DebugOut_SBCFloat(float* a_outfData);
        [DllImport("innoSDK.dll", SetLastError = true, CharSet = CharSet.Unicode)]
        internal static extern unsafe System.UInt32 inno_DebugOut_SBCFloat(System.Single[]
a_outfData);

        //       DWORD PASCAL EXPORT Motor_SetLeftWheelSpeed(WORD a_InSpeed);
        [DllImport("innoSDK.dll", SetLastError = true, CharSet = CharSet.Unicode)]
        internal static extern System.UInt32 Motor_SetLeftWheelSpeed(System.UInt16 a_InSpeed);

        //       DWORD PASCAL EXPORT Motor_SetRightWheelSpeed(WORD a_InSpeed);
        [DllImport("innoSDK.dll", SetLastError = true, CharSet = CharSet.Unicode)]
        internal static extern System.UInt32 Motor_SetRightWheelSpeed(System.UInt16 a_InSpeed);

        //       BOOL PASCAL EXPORT Sonar_GetTCRT5000Status(unsigned char* a_OutData);
        [DllImport("innoSDK.dll", SetLastError = true, CharSet = CharSet.Unicode)]
        internal static extern unsafe System.Int32 Sonar_GetTCRT5000Status(System.Byte[] a_OutData);

        //       DWORD PASCAL EXPORT Sonar_Ranging(unsigned char a_inChannel);
        [DllImport("innoSDK.dll", SetLastError = true, CharSet = CharSet.Unicode)]
        internal static extern System.UInt32 Sonar_Ranging(System.Byte a_inChannel);

        //       DWORD PASCAL EXPORT Sonar_GetDistance(unsigned char a_inChannel, unsigned char*
a_poutStatus, unsigned short* a_poutWORD);
        [DllImport("innoSDK.dll", SetLastError = true, CharSet = CharSet.Unicode)]
        internal static extern unsafe System.UInt32 Sonar_GetDistance(System.Byte a_inChannel,
System.Byte[] a_poutStatus, System.UInt16[] a_poutWORD);
    }
}
```

innoBotServiceTypes.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
```

```
92  using Microsoft.Ccr.Core;
93  using Microsoft.Dss.Core.Attributes;
94  using Microsoft.Dss.ServiceModel.Dssp;
95  using Microsoft.Dss.ServiceModel.DsspServiceBase;
96  using W3C.Soap;
97
98  namespace innoBotService
99  {
100     /// <summary>
101     /// innoBotService contract class
102     /// </summary>
103     public sealed class Contract
104     {
105         /// <summary>
106         /// DSS contract identifer for innoBotService
107         /// </summary>
108         [DataMember]
109         public const string Identifier = "http://schemas.tempuri.org/2010/03/innobotservice.html";
110     }
111
112     /// <summary>
113     /// innoBotService state
114     /// </summary>
115     [DataContract]
116     public class innoBotServiceState
117     {
118     }
119
120     #region New ServicePort: MSRDS中Service的溝通Port
121     /// <summary>
122     /// innoBotService main operations port
123     /// </summary>
124     [ServicePort]
125     public class innoBotServiceOperations : PortSet<
126         DsspDefaultLookup,
127         DsspDefaultDrop,
128         Get,
129         Subscribe,
130         Connect,
131         Disconnect,
```

```
132             IsConnect,
133             ResetTarget,
134             Sonar_SetPosition,
135             Sonar_SetAngle,
136             Motor_SetLeftWheelSpeed,
137             Motor_SetRightWheelSpeed,
138             Sonar_GetTCRT5000Status,
139             Sonar_Ranging,
140             Sonar_GetDistance,
141             SetDrivePower,
142             Sonar_AngleDistance
143             >
144         {
145         }
146         #endregion
147
148         /// <summary>
149         /// innoBotService get operation
150         /// </summary>
151         public class Get : Get<GetRequestType, PortSet<innoBotServiceState, Fault>>
152         {
153             /// <summary>
154             /// Creates a new instance of Get
155             /// </summary>
156             public Get()
157             {
158             }
159
160             /// <summary>
161             /// Creates a new instance of Get
162             /// </summary>
163             /// <param name="body">the request message body</param>
164             public Get(GetRequestType body)
165                 : base(body)
166             {
167             }
168
169             /// <summary>
170             /// Creates a new instance of Get
171             /// </summary>
```

```
172         /// <param name="body">the request message body</param>
173         /// <param name="responsePort">the response port for the request</param>
174         public Get(GetRequestType body, PortSet<innoBotServiceState, Fault> responsePort)
175             : base(body, responsePort)
176         {
177         }
178     }
179 
180     /// <summary>
181     /// innoBotService subscribe operation
182     /// </summary>
183     public class Subscribe : Subscribe<SubscribeRequestType, PortSet<SubscribeResponseType, Fault>>
184     {
185         /// <summary>
186         /// Creates a new instance of Subscribe
187         /// </summary>
188         public Subscribe()
189         {
190         }
191 
192         /// <summary>
193         /// Creates a new instance of Subscribe
194         /// </summary>
195         /// <param name="body">the request message body</param>
196         public Subscribe(SubscribeRequestType body)
197             : base(body)
198         {
199         }
200 
201         /// <summary>
202         /// Creates a new instance of Subscribe
203         /// </summary>
204         /// <param name="body">the request message body</param>
205         /// <param name="responsePort">the response port for the request</param>
206         public Subscribe(SubscribeRequestType body, PortSet<SubscribeResponseType, Fault>
207 responsePort)
208             : base(body, responsePort)
209         {
210         }
211     }
```

```
212
213     #region New DataContract: 各Port所需輸入、出變數之設定
214
215     [DataContract]
216     public class Connect_Input
217     {
218     }
219     [DataContract]
220     public class Connect_Output
221     {
222         [DataMember]
223         public bool Result { get; set; }
224     }
225     [DisplayName("Connect")]
226     [Description("bool inno_Connect()")]
227     public class Connect : Submit<Connect_Input, PortSet<Connect_Output, Fault>>
228     {
229     }
230
231     [DataContract]
232     public class Disconnect_Input
233     {
234     }
235     [DataContract]
236     public class Disconnect_Output
237     {
238     }
239     [DisplayName("Disconnect")]
240     [Description("void inno_Disconnect()")]
241     public class Disconnect : Submit<Disconnect_Input, PortSet<Disconnect_Output, Fault>>
242     {
243     }
244
245     [DataContract]
246     public class IsConnect_Input
247     {
248     }
249     [DataContract]
250     public class IsConnect_Output
251     {
```

```
252             [DataMember]
253             public bool Result { get; set; }
254         }
255         [DisplayName("IsConnect")]
256         [Description("bool inno_IsConnect()")]
257         public class IsConnect : Submit<IsConnect_Input, PortSet<IsConnect_Output, Fault>>
258         {
259         }
260 
261         [DataContract]
262         public class ResetTarget_Input
263         {
264         }
265         [DataContract]
266         public class ResetTarget_Output
267         {
268             [DataMember]
269             public UInt32 Result { get; set; }
270         }
271         [DisplayName("ResetTarget")]
272         [Description("System.UInt32 inno_ResetTarget()")]
273         public class ResetTarget : Submit<ResetTarget_Input, PortSet<ResetTarget_Output, Fault>>
274         {
275         }
276 
277         [DataContract]
278         public class Sonar_SetPosition_Input
279         {
280             [DataMember]
281             public UInt16 SonarPosition { get; set; }
282         }
283         [DataContract]
284         public class Sonar_SetPosition_Output
285         {
286             [DataMember]
287             public UInt32 Result { get; set; }
288         }
289         [DisplayName("Sonar_SetPosition")]
290         [Description("System.UInt32 Sonar_SetPosition(System.UInt16 a_InPos)")]
291         public class Sonar_SetPosition : Submit<Sonar_SetPosition_Input, PortSet<Sonar_SetPosition_Output,
```

```
Fault>>
    {
    }

    [DataContract]
    public class Sonar_SetAngle_Input
    {
        [DataMember]
        public double SonarAngle { get; set; }
    }
    [DataContract]
    public class Sonar_SetAngle_Output
    {
        [DataMember]
        public UInt32 Result { get; set; }
    }
    [DisplayName("Sonar_SetAngle")]
    [Description("System.UInt32 Sonar_SetAngle(System.UInt16 a_InPos)")]
    public class Sonar_SetAngle : Submit<Sonar_SetAngle_Input, PortSet<Sonar_SetAngle_Output, Fault>>
    {
    }

    [DataContract]
    public class Motor_SetLeftWheelSpeed_Input
    {
        [DataMember]
        public UInt16 LeftWheelSpeed { get; set; }
    }
    [DataContract]
    public class Motor_SetLeftWheelSpeed_Output
    {
        [DataMember]
        public UInt32 Result { get; set; }
    }
    [DisplayName("Motor_SetLeftWheelSpeed")]
    [Description("System.UInt32 Motor_SetLeftWheelSpeed(System.UInt16 a_InSpeed)")]
    public class Motor_SetLeftWheelSpeed : Submit<Motor_SetLeftWheelSpeed_Input,
PortSet<Motor_SetLeftWheelSpeed_Output, Fault>>
    {
    }
```

```
332
333         [DataContract]
334         public class Motor_SetRightWheelSpeed_Input
335         {
336             [DataMember]
337             public UInt16 RightWheelSpeed { get; set; }
338         }
339         [DataContract]
340         public class Motor_SetRightWheelSpeed_Output
341         {
342             [DataMember]
343             public UInt32 Result { get; set; }
344         }
345         [DisplayName("Motor_SetRightWheelSpeed")]
346         [Description("System.UInt32 Motor_SetRightWheelSpeed(System.UInt16 a_InSpeed)")]
347         public class Motor_SetRightWheelSpeed : Submit<Motor_SetRightWheelSpeed_Input,
348     PortSet<Motor_SetRightWheelSpeed_Output, Fault>>
349         {
350         }
351
352         [DataContract]
353         public class Sonar_GetTCRT5000Status_Input
354         {
355         }
356         [DataContract]
357         public class Sonar_GetTCRT5000Status_Output
358         {
359             [DataMember]
360             public Int32 Result { get; set; }
361             [DataMember]
362             public Byte[] TotalStatus { get; set; }
363             [DataMember]
364             public int[] EachStatus { get; set; }
365         }
366         [DisplayName("Sonar_GetTCRT5000Status")]
367         [Description("unsafe System.Int32 Sonar_GetTCRT5000Status(System.Byte[] a_OutData)")]
368         public class Sonar_GetTCRT5000Status : Submit<Sonar_GetTCRT5000Status_Input,
369     PortSet<Sonar_GetTCRT5000Status_Output, Fault>>
370         {
371         }
```

```
372
373         [DataContract]
374         public class Sonar_Ranging_Input
375         {
376             [DataMember]
377             public int SonarChannel { get; set; }
378         }
379         [DataContract]
380         public class Sonar_Ranging_Output
381         {
382             [DataMember]
383             public UInt32 Result { get; set; }
384         }
385         [DisplayName("Sonar_Ranging")]
386         [Description("System.UInt32 Sonar_Ranging(System.Byte a_inChannel)")]
387         public class Sonar_Ranging : Submit<Sonar_Ranging_Input, PortSet<Sonar_Ranging_Output, Fault>>
388         {
389         }
390
391         [DataContract]
392         public class Sonar_GetDistance_Input
393         {
394             [DataMember]
395             public int SonarChannel { get; set; }
396         }
397         [DataContract]
398         public class Sonar_GetDistance_Output
399         {
400             [DataMember]
401             public UInt32 Result { get; set; }
402             [DataMember]
403             public Byte[] Status { get; set; }
404             [DataMember]
405             public UInt16[] Distance { get; set; }
406         }
407         [DisplayName("Sonar_GetDistance")]
408         [Description("unsafe System.UInt32 Sonar_GetDistance(System.Byte a_inChannel, System.Byte[]
409     a_poutStatus, System.UInt16[] a_poutWORD)")]
410         public class Sonar_GetDistance : Submit<Sonar_GetDistance_Input, PortSet<Sonar_GetDistance_Output,
411     Fault>>
```

```
412         {
413         }
414
415         [DataContract]
416         public class SetDrivePower_Input
417         {
418             [DataMember]
419             public double LeftWheelPower { get; set; }
420             [DataMember]
421             public double RightWheelPower { get; set; }
422         }
423         [DataContract]
424         public class SetDrivePower_Output
425         {
426             [DataMember]
427             public UInt32 LeftResult { get; set; }
428             [DataMember]
429             public UInt32 RightResult { get; set; }
430         }
431         [DisplayName("SetDrivePower")]
432         [Description("System.UInt32 SetDrivePower(dboule Power)")]
433         public class SetDrivePower : Submit<SetDrivePower_Input, PortSet<SetDrivePower_Output, Fault>>
434         {
435         }
436
437         [DataContract]
438         public class Sonar_AngleDistance_Input
439         {
440             [DataMember]
441             public int SonarChannel { get; set; }
442             [DataMember]
443             public double SonarAngle { get; set; }
444         }
445         [DataContract]
446         public class Sonar_AngleDistance_Output
447         {
448             [DataMember]
449             public UInt32 AngleResult { get; set; }
450             [DataMember]
451             public UInt32 RangingResult { get; set; }
```

```
452             [DataMember]
453             public UInt32 DistanceResult { get; set; }
454             [DataMember]
455             public Byte[] Status { get; set; }
456             [DataMember]
457             public UInt16[] Distance { get; set; }
458         }
459         [DisplayName("Sonar_AngleDistance")]
460         [Description("unsafe System.UInt32 Sonar_AngleDistance(System.Byte a_inChannel, double Angle)")]
461         public class Sonar_AngleDistance : Submit<Sonar_AngleDistance_Input,
462     PortSet<Sonar_AngleDistance_Output, Fault>>
463         {
464         }
465 
466         #endregion
467 
468     }
```

innoBotService.cs

```
469 using System;
470 using System.Collections.Generic;
471 using System.ComponentModel;
472 using Microsoft.Ccr.Core;
473 using Microsoft.Dss.Core.Attributes;
474 using Microsoft.Dss.ServiceModel.Dssp;
475 using Microsoft.Dss.ServiceModel.DsspServiceBase;
476 using W3C.Soap;
477 using submgr = Microsoft.Dss.Services.SubscriptionManager;
478 // For Window Form, Microsoft.Ccr.Adapters.WinForms
479 using Microsoft.Ccr.Adapters.WinForms;
480 using System.Threading;
481 
482 namespace innoBotService
483 {
484     [Contract(Contract.Identifier)]
485     [DisplayName("innoBotService")]
486     [Description("innoBotService service (no description provided)")]
487     class innoBotService : DsspServiceBase
488 
```

```
{
    /// <summary>
    /// Service state
    /// </summary>
    [ServiceState]
    innoBotServiceState _state = new innoBotServiceState();

    /// <summary>
    /// Main service port
    /// </summary>
    [ServicePort("/innoBotService", AllowMultipleInstances = true)]
    innoBotServiceOperations _mainPort = new innoBotServiceOperations();

    [SubscriptionManagerPartner]
    submgr.SubscriptionManagerPort _submgrPort = new submgr.SubscriptionManagerPort();

    /// <summary>
    /// Service constructor
    /// </summary>
    public innoBotService(DsspServiceCreationPort creationPort)
        : base(creationPort)
    {
    }

    #region New Variable
    static ControlPanel _ControlPanel;
    #endregion

    /// <summary>
    /// Service start
    /// </summary>
    protected override void Start()
    {

        //
        // Add service specific initialization here
        //

        base.Start();

```

```
529             //WinFormsServicePort.Post(new RunForm(CreateForm));
530             #region Run Win Form
531             WinFormsServicePort.Post(new RunForm(() =>
532             {
533                 _ControlPanel = new ControlPanel();
534                 _ControlPanel.Show();
535                 return _ControlPanel;
536             }));
537             #endregion
538         }
539 
540         /// <summary>
541         /// Handles Subscribe messages
542         /// </summary>
543         /// <param name="subscribe">the subscribe request</param>
544         [ServiceHandler]
545         public void SubscribeHandler(Subscribe subscribe)
546         {
547             SubscribeHelper(_submgrPort, subscribe.Body, subscribe.ResponsePort);
548         }
549 
550 
551         #region New ServiceHandler: MSRDS輸入資料到Port時對應之處理方式
552 
553         [ServiceHandler(ServiceHandlerBehavior.Concurrent)]
554         public IEnumerator<ITask> Connect_Handler(Connect Connect_Var)
555         {
556             var Result = innoBotControl.inno_Connect();
557             _ControlPanel.UpdateLabel("Connect Result: " + Result.ToString());
558             Connect_Var.ResponsePort.Post(new Connect_Output() { Result = Result });
559             yield break;
560         }
561 
562         [ServiceHandler(ServiceHandlerBehavior.Concurrent)]
563         public IEnumerator<ITask> Disconnect_Handler(Disconnect Disconnect_Var)
564         {
565             innoBotControl.inno_Disconnect();
566             _ControlPanel.UpdateLabel("Disconnect");
567             yield break;
568         }
```

```
569
570         [ServiceHandler(ServiceHandlerBehavior.Concurrent)]
571         public IEnumerator<ITask> IsConnect_Handler(IsConnect IsConnect_Var)
572         {
573             var Result = innoBotControl.inno_IsConnect();
574             _ControlPanel.UpdateLabel("IsConnect Result: " + Result.ToString());
575             IsConnect_Var.ResponsePort.Post(new IsConnect_Output() { Result = Result });
576             yield break;
577         }
578
579         [ServiceHandler(ServiceHandlerBehavior.Concurrent)]
580         public IEnumerator<ITask> ResetTarget_Handler(ResetTarget ResetTarget_Var)
581         {
582             var Result = innoBotControl.inno_ResetTarget();
583             _ControlPanel.UpdateLabel("ResetTarget Result: " + Result.ToString());
584             ResetTarget_Var.ResponsePort.Post(new ResetTarget_Output() { Result = Result });
585             yield break;
586         }
587
588         [ServiceHandler(ServiceHandlerBehavior.Concurrent)]
589         public IEnumerator<ITask> Sonar_SetPosition_Handler(Sonar_SetPosition Sonar_SetPosition_Var)
590         {
591             var Result = innoBotControl.Sonar_SetPosition(Sonar_SetPosition_Var.Body.SonarPosition);
592             _ControlPanel.UpdateLabel("Sonar_SetPosition: " +
593 Sonar_SetPosition_Var.Body.SonarPosition.ToString() + ", Result: " + Result.ToString());
594             Sonar_SetPosition_Var.ResponsePort.Post(new Sonar_SetPosition_Output() { Result =
595 Result });
596             yield break;
597         }
598
599         [ServiceHandler(ServiceHandlerBehavior.Concurrent)]
600         public IEnumerator<ITask> Sonar_SetAngle_Handler(Sonar_SetAngle Sonar_SetAngle_Var)
601         {
602             double SonarAngle = Sonar_SetAngle_Var.Body.SonarAngle;
603             if (SonarAngle < -60)
604             {
605                 SonarAngle = -60;
606             }
607             else if (60 < SonarAngle)
608             {
```

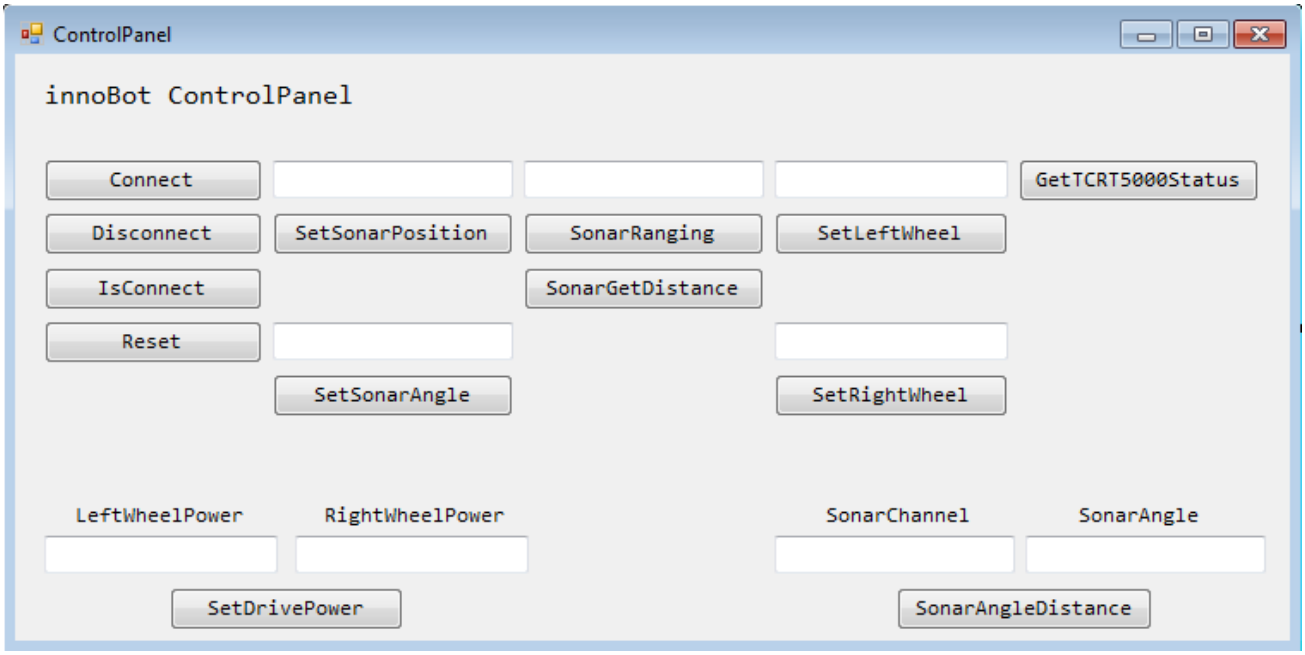
```
609                 SonarAngle = 60;
610             }
611             var SonarPosition = Convert.ToUInt16(1500 - SonarAngle / 90 * 800);
612             var Result = innoBotControl.Sonar_SetPosition(SonarPosition);
613             _ControlPanel.UpdateLabel("Sonar_SetAngle: " +
614 Sonar_SetAngle_Var.Body.SonarAngle.ToString() + ", Result: " + Result.ToString());
615             Sonar_SetAngle_Var.ResponsePort.Post(new Sonar_SetAngle_Output() { Result = Result });
616             yield break;
617         }
618 
619         [ServiceHandler(ServiceHandlerBehavior.Concurrent)]
620         public IEnumerator<ITask> Motor_SetLeftWheelSpeed_Handler(Motor_SetLeftWheelSpeed
621 Motor_SetLeftWheelSpeed_Var)
622         {
623             var Result =
624 innoBotControl.Motor_SetLeftWheelSpeed(Motor_SetLeftWheelSpeed_Var.Body.LeftWheelSpeed);
625             _ControlPanel.UpdateLabel("Motor_SetLeftWheelSpeed: " +
626 Motor_SetLeftWheelSpeed_Var.Body.LeftWheelSpeed.ToString() + ", Result: " + Result.ToString());
627             Motor_SetLeftWheelSpeed_Var.ResponsePort.Post(new Motor_SetLeftWheelSpeed_Output()
628 { Result = Result });
629             yield break;
630         }
631 
632         [ServiceHandler(ServiceHandlerBehavior.Concurrent)]
633         public IEnumerator<ITask> Motor_SetRightWheelSpeed_Handler(Motor_SetRightWheelSpeed
634 Motor_SetRightWheelSpeed_Var)
635         {
636             var Result =
637 innoBotControl.Motor_SetRightWheelSpeed(Motor_SetRightWheelSpeed_Var.Body.RightWheelSpeed);
638             _ControlPanel.UpdateLabel("Motor_SetRightWheelSpeed: " +
639 Motor_SetRightWheelSpeed_Var.Body.RightWheelSpeed.ToString() + ", Result: " + Result.ToString());
640             Motor_SetRightWheelSpeed_Var.ResponsePort.Post(new Motor_SetRightWheelSpeed_Output()
641 { Result = Result });
642             yield break;
643         }
644 
645         [ServiceHandler(ServiceHandlerBehavior.Concurrent)]
646         public IEnumerator<ITask> Sonar_GetTCRT5000Status_Handler(Sonar_GetTCRT5000Status
647 Sonar_GetTCRT5000Status_Var)
648         {
```

```
            Byte[] TotalStatus = new Byte[1];
            var Result = innoBotControl.Sonar_GetTCRT5000Status(TotalStatus);
            int Status = (int)TotalStatus[0];
            int[] EachStatus = new int[5];
            EachStatus[4] = Status / 16;
            Status = Status - EachStatus[4] * 16;
            EachStatus[3] = Status / 8;
            Status = Status - EachStatus[3] * 8;
            EachStatus[2] = Status / 4;
            Status = Status - EachStatus[2] * 4;
            EachStatus[1] = Status / 2;
            Status = Status - EachStatus[1] * 2;
            EachStatus[0] = Status / 1;
            _ControlPanel.UpdateLabel("Sonar_GetTCRT5000Status Result: " + Result.ToString() + ", " +
TotalStatus[0].ToString() + " : " + EachStatus[0] + EachStatus[1] + EachStatus[2] + EachStatus[3] +
EachStatus[4]);
            Sonar_GetTCRT5000Status_Var.ResponsePort.Post(new Sonar_GetTCRT5000Status_Output()
{ Result = Result, TotalStatus = TotalStatus, EachStatus = EachStatus });
            yield break;
        }

        [ServiceHandler(ServiceHandlerBehavior.Concurrent)]
        public IEnumerator<ITask> Sonar_Ranging_Handler(Sonar_Ranging Sonar_Ranging_Var)
        {
            var Result =
innoBotControl.Sonar_Ranging(Convert.ToByte(Sonar_Ranging_Var.Body.SonarChannel));
            _ControlPanel.UpdateLabel("Sonar_Ranging: " +
Sonar_Ranging_Var.Body.SonarChannel.ToString() + ", Result: " + Result.ToString());
            Sonar_Ranging_Var.ResponsePort.Post(new Sonar_Ranging_Output() { Result = Result });
            yield break;
        }

        [ServiceHandler(ServiceHandlerBehavior.Concurrent)]
        public IEnumerator<ITask> Sonar_GetDistance_Handler(Sonar_GetDistance Sonar_GetDistance_Var)
        {
            Byte[] Status = new Byte[1];
            UInt16[] Distance = new UInt16[1];
            var Result =
innoBotControl.Sonar_GetDistance(Convert.ToByte(Sonar_GetDistance_Var.Body.SonarChannel), Status,
Distance);
```

```
689             _ControlPanel.UpdateLabel("Sonar_GetDistance: " +
690 Sonar_GetDistance_Var.Body.SonarChannel.ToString() + ", Result: " + Result.ToString() + ", Status: "
691 + Status[0].ToString() + ", Distance: " + Distance[0].ToString() + " cm");
692             Sonar_GetDistance_Var.ResponsePort.Post(new Sonar_GetDistance_Output() { Result = Result,
693 Status = Status, Distance = Distance });
694             yield break;
695         }
696 
697         [ServiceHandler(ServiceHandlerBehavior.Concurrent)]
698         public IEnumerator<ITask> SetDrivePower_Handler(SetDrivePower SetDrivePower_Var)
699         {
700             var LeftWheelPower = SetDrivePower_Var.Body.LeftWheelPower;
701             var RightWheelPower = SetDrivePower_Var.Body.RightWheelPower;
702             var LeftWheelSpeed = Convert.ToUInt16(1500 + LeftWheelPower * 500);
703             var RightWheelSpeed = Convert.ToUInt16(1500 - RightWheelPower * 500);
704             var LeftResult = innoBotControl.Motor_SetLeftWheelSpeed(LeftWheelSpeed);
705             var RightResult = innoBotControl.Motor_SetRightWheelSpeed(RightWheelSpeed);
706             _ControlPanel.UpdateLabel("SetDrivePower; Left Power: " + LeftWheelPower.ToString() + ",
707 Result: " + LeftResult.ToString() + "; Right Power: " + RightWheelPower.ToString() + ", Result: " +
708 RightResult.ToString());
709             SetDrivePower_Var.ResponsePort.Post(new SetDrivePower_Output() { LeftResult = LeftResult,
710 RightResult = RightResult});
711             yield break;
712         }
713 
714         [ServiceHandler(ServiceHandlerBehavior.Concurrent)]
715         public IEnumerator<ITask> Sonar_AngleDistance_Handler(Sonar_AngleDistance
716 Sonar_AngleDistance_Var)
717         {
718             var SonarChannel = Convert.ToByte(Sonar_AngleDistance_Var.Body.SonarChannel);
719             var SonarAngle = Sonar_AngleDistance_Var.Body.SonarAngle;
720             if (SonarAngle < -60)
721             {
722                 SonarAngle = -60;
723             }
724             else if (60 < SonarAngle)
725             {
726                 SonarAngle = 60;
727             }
728             var SonarPosition = Convert.ToUInt16(1500 - SonarAngle / 90 * 800);
```

```
729             var PositionResult = innoBotControl.Sonar_SetPosition(SonarPosition);
730             Thread.Sleep(200);
731 
732             var RangingResult = innoBotControl.Sonar_Ranging(SonarChannel);
733             Byte[] Status = new Byte[1];
734             UInt16[] Distance = new UInt16[1];
735             var DistanceResult = innoBotControl.Sonar_GetDistance(SonarChannel, Status, Distance);
736             Console.WriteLine(Distance);
737             _ControlPanel.UpdateLabel("Sonar_AngleDistance; Angle: " + SonarAngle.ToString() + ",
738 Distance: " + Distance[0].ToString() + ", Status: " + Status[0].ToString() + ", Result: " +
739 PositionResult.ToString() + ", " + RangingResult.ToString() + ", " + DistanceResult.ToString());
740             Sonar_AngleDistance_Var.ResponsePort.Post(new Sonar_AngleDistance_Output() { Distance =
741 Distance, Status = Status, AngleResult = PositionResult, RangingResult = RangingResult, DistanceResult
742 = DistanceResult });
743             yield break;
744         }
745 
746         #endregion
747 
748 
749 
750     }
751 }
```

ControlPanel



ControlPanel.cs

```
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;
// Add
using System.Threading;

namespace innoBotService
{
    public partial class ControlPanel : Form
    {
        public ControlPanel()
        {
            InitializeComponent();
        }

```

```
775         #region New ControlPanel Function
776
777         public void UpdateLabel(string Status)
778         {
779             Label.Text = Status;
780         }
781
782         private void ConnectButton_Click(object sender, EventArgs e)
783         {
784             var Result = innoBotControl.inno_Connect();
785             UpdateLabel("Connect Result: " + Result.ToString());
786         }
787
788         private void DisconnectButton_Click(object sender, EventArgs e)
789         {
790             innoBotControl.inno_Disconnect();
791             UpdateLabel("Disconnect");
792         }
793
794         private void IsConnectButton_Click(object sender, EventArgs e)
795         {
796             var Result = innoBotControl.inno_IsConnect();
797             UpdateLabel("IsConnect Result: " + Result.ToString());
798         }
799
800         private void ResetButton_Click(object sender, EventArgs e)
801         {
802             var Result = innoBotControl.inno_ResetTarget();
803             UpdateLabel("ResetTarget Result: " + Result.ToString());
804         }
805
806         private void SetSonarPositionButton_Click(object sender, EventArgs e)
807         {
808             var Result =
809 innoBotControl.Sonar_SetPosition(Convert.ToUInt16(SetSonarPositionTextBox.Text));
810             UpdateLabel("Sonar_SetPosition: " + SetSonarPositionTextBox.Text + ", Result: " +
811 Result.ToString());
812         }
813
814         private void SetSonarAngleButton_Click(object sender, EventArgs e)
```

```
815         {
816             var SonarAngle = Convert.ToDouble(SetSonarAngleTextBox.Text);
817             if (SonarAngle < -60)
818             {
819                 SonarAngle = -60;
820             }
821             else if (60 < SonarAngle)
822             {
823                 SonarAngle = 60;
824             }
825             var SonarPosition = Convert.ToUInt16(1500 - SonarAngle / 90 * 800);
826             var PositionResult = innoBotControl.Sonar_SetPosition(SonarPosition);
827         }
828 
829         private void Motor_SetLeftWheelButton_Click(object sender, EventArgs e)
830         {
831             var Result =
832 innoBotControl.Motor_SetLeftWheelSpeed(Convert.ToUInt16(Motor_SetLeftWheelTextBox.Text));
833             UpdateLabel("Motor_SetLeftWheelSpeed: " + Motor_SetLeftWheelTextBox.Text + ", Result: " +
834 Result.ToString());
835         }
836 
837         private void Motor_SetRightWheelButton_Click(object sender, EventArgs e)
838         {
839             var Result =
840 innoBotControl.Motor_SetRightWheelSpeed(Convert.ToUInt16(Motor_SetRightWheelTextBox.Text));
841             UpdateLabel("Motor_SetRightWheelSpeed: " + Motor_SetRightWheelTextBox.Text + ", Result: "
842 + Result.ToString());
843         }
844 
845         private void Sonar_GetTCRT5000StatusButton_Click(object sender, EventArgs e)
846         {
847             Byte[] TotalStatus = new Byte[1];
848             var Result = innoBotControl.Sonar_GetTCRT5000Status(TotalStatus);
849             int Status = (int)TotalStatus[0];
850             int[] EachStatus = new int[5];
851             EachStatus[4] = Status / 16;
852             Status = Status - EachStatus[4] * 16;
853             EachStatus[3] = Status / 8;
854             Status = Status - EachStatus[3] * 8;
```

```
            EachStatus[2] = Status / 4;
            Status = Status - EachStatus[2] * 4;
            EachStatus[1] = Status / 2;
            Status = Status - EachStatus[1] * 2;
            EachStatus[0] = Status / 1;
            UpdateLabel("Sonar_GetTCRT5000Status Result: " + Result.ToString() + ", " +
TotalStatus[0].ToString() + " : " + EachStatus[0] + EachStatus[1] + EachStatus[2] + EachStatus[3] +
EachStatus[4]);
        }

        private void Sonar_RangingButton_Click(object sender, EventArgs e)
        {
            var Result = innoBotControl.Sonar_Ranging(Convert.ToByte(SonarChannelTextBox.Text));
            UpdateLabel("Sonar_Ranging: " + SonarChannelTextBox.Text + ", Result: " +
Result.ToString());
        }

        private void Sonar_GetDistanceButton_Click(object sender, EventArgs e)
        {
            Byte[] Status = new Byte[1];
            UInt16[] Distance = new UInt16[1];
            var Result = innoBotControl.Sonar_GetDistance(Convert.ToByte(SonarChannelTextBox.Text),
Status, Distance);
            UpdateLabel("Sonar_GetDistance: " + SonarChannelTextBox.Text + ", Result: " +
Result.ToString() + ", Status: " + Status[0].ToString() + ", Distance: " + Distance[0].ToString() +
" cm");
        }

        private void SetDrivePowerButton_Click(object sender, EventArgs e)
        {
            var LeftWheelPower = Convert.ToDouble(LeftWheelPowerTextBox.Text);
            var RightWheelPower = Convert.ToDouble(RightWheelPowerTextBox.Text);
            var LeftWheelSpeed = Convert.ToUInt16(1500 + LeftWheelPower * 500);
            var RightWheelSpeed = Convert.ToUInt16(1500 - RightWheelPower * 500);
            var LeftResult = innoBotControl.Motor_SetLeftWheelSpeed(LeftWheelSpeed);
            var RightResult = innoBotControl.Motor_SetRightWheelSpeed(RightWheelSpeed);
            UpdateLabel("SetDrivePower; Left Power: " + LeftWheelPowerTextBox.Text + ", Result: " +
LeftResult.ToString() + "; Right Power: " + RightWheelPowerTextBox.Text + ", Result: " +
RightResult.ToString());
        }
```

```
895
896         private void Sonar_AngleDistanceButton_Click(object sender, EventArgs e)
897         {
898             var SonarChannel = Convert.ToByte(ChannelTextBox.Text);
899             var SonarAngle = Convert.ToDouble(AngleTextBox.Text);
900             if (SonarAngle < -60)
901             {
902                 SonarAngle = -60;
903             }
904             else if (60 < SonarAngle)
905             {
906                 SonarAngle = 60;
907             }
908             var SonarPosition = Convert.ToUInt16(1500 - SonarAngle / 90 * 800);
909             var PositionResult = innoBotControl.Sonar_SetPosition(SonarPosition);
910             Thread.Sleep(200);
911 
912             var RangingResult = innoBotControl.Sonar_Ranging(SonarChannel);
913             Byte[] Status = new Byte[1];
914             UInt16[] Distance = new UInt16[1];
915             var DistanceResult = innoBotControl.Sonar_GetDistance(SonarChannel, Status, Distance);
916             UpdateLabel("Sonar_AngleDistance; Angle: " + SonarAngle.ToString() + ", Distance: " +
917 Distance[0].ToString() + ", Status: " + Status[0].ToString() + ", Result: " + PositionResult.ToString()
918 + ", " + RangingResult.ToString() + ", " + DistanceResult.ToString());
919         }
920 
921         #endregion
922     }
923 }
```