Manual

FlexiVision

OMRON PLUG-IN

ars | Feeding Industrial Robotics

INDICE

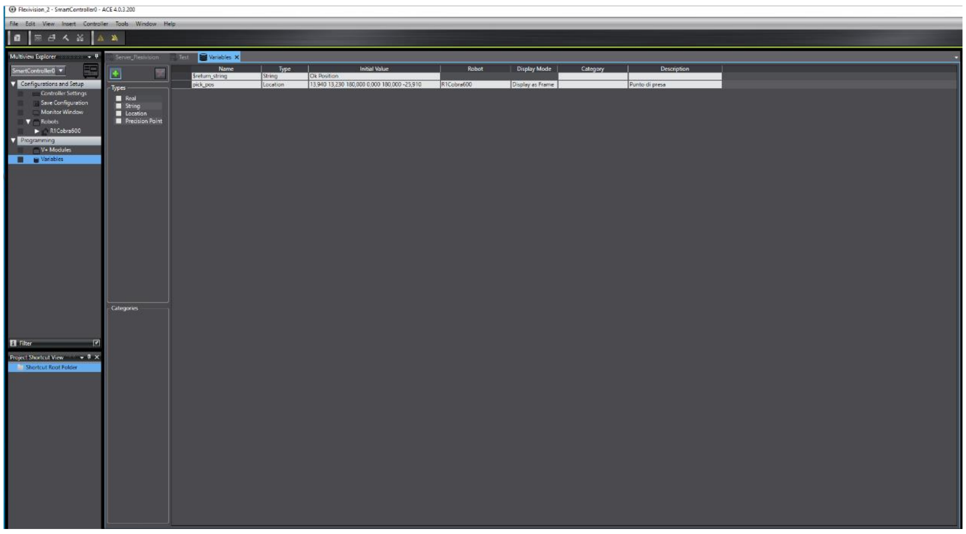
Questo Plugin è nato con l'idea di comunicare in maniera **rapida e sicura con il FlexiBowl®** tramite i robot **Omron**, mediante il **software Omron Ace** versione 4 o superiore.

Il Plugin non necessita di licenze aggiuntive Omron.

FlexiBowl® Plug-In

Installazione Plug-In

Step 1.

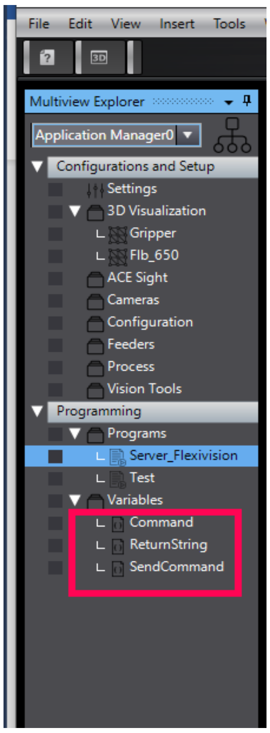


Name	Type	Initial Value	Robot	Display Mode	Category	Description
$return_string	String	Ok Position				
pick_pos	Location	13,940 13,230 180,000 0,000 180,000 -25,910	R1Cobra600	Display as Frame		Punto di presa

Creare due variabili V+, una per la **locazione** e una per **la stringa di risposta**:

- $return_string
- pick_pos

Step 2.



Creare tre variabili c# nell'Application Manager:

- ***Command*** è una stringa dove verrà indicata l'operazione da eseguire su Flexivision;
- ***ReturnString*** è una stringa che contiene la risposta di Flexivision;
- ***SendCommand*** è una variabile numerica, che serve da semaforo, quando è impostata a 1 viene interpretato il comando da inviare a Flexivision, una volta finito di processare il comando, questa variabile tornerà a 0

Installazione Plug-In

Step 3.

Ora creiamo uno script c#, dove andrà settato l'Ip del server che in questo caso sarà il robot, e la porta di comunicazione.

```
using Ace.Server.Core.Variable;
using Ace.Server.Adept.Controllers;
using Ace.Services.NameLookup;
using Ace.Server;
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.Threading.Tasks;
using Ace.Server.Adept.Controllers.Link;
using Ace.Communication.Services.Link;

namespace Ace.Custom {
        public class Program {
                public INameLookupService ace;
                        // Incoming data from the client.
                        public static string data = null;
                        public void Main() {
                                string ip = "192.168.1.3";  int port = = 8887;
                                // Client socket.  Socket handler = null;
                                bool Violated = false;
                                IVariableString Command = (IVariableString) ace["/Application
  Manager0/Variables/Command"];
                                IVariableString ReturnString = (IVariableString) ace["/Application
  Manager0/Variables/ReturnString"];
                                IVariableNumeric SendCommand = (IVariableNumeric)
  ace["/Application  Manager0/Variables/SendCommand"];
                                IAdeptController adeptController = (IAdeptController)
  ace["/SmartController0/Controller Settings"];
```

Installazione Plug-In

```
// Get access to the communications link

IVpLink link = adeptController.Link;
 // Data buffer for incoming data.
 byte[] bytes = new Byte[1024];
 IPAddress ipAddress = IPAddress.Parse(ip);
 IPEndPoint localEndPoint = new IPEndPoint(ipAddress, port);

 // Create a TCP/IP socket.
 Socket listener = new Socket(ipAddress.AddressFamily, SocketType.Stream, ProtocolType.Tcp);

 // Bind the socket to the local endpoint and
 // listen for incoming connections.
 try {
        listener.Bind(localEndPint);  listener.Listen(10);
        Trace.WriteLine("Waiting for a connection...");
        // Program is suspended while waiting for an incoming connection.
        handler = listener.Accept();
        Trace.WriteLine("Flexivision connected...");
        // Start listening for connections.
        while (true) {
               Trace.WriteLine("Waiting for command");
 ReturnString.CurrentValue = "";
               Violated = false;
               while (Violated != true) {
                      if (SendCommand.CurrentValue == 1)  Violated = true;
               }
        data = null;
        //I analyze the request
        Trace.WriteLine("Command: " + Command.CurrentValue);
        //set_Recipe
        if (Command.CurrentValue.Contains("set_Recipe")) {
               // Convert the string data to byte data using ASCII encoding.
               byte[] byteData
 =mEncoding.ASCII.GetBytes(Command.CurrentValue);
               handler.Send(byteData);
               ReturnString.CurrentValue = "Ok";
               // Set, read, and delete a string variable
               link.SetS("$return_string",ReturnString.CurrentValue);
               SendCommand.CurrentValue = -1;
               }
        //get_Recipe
        if (Command.CurrentValue.Contains("get_Recipe")) {
               // Convert the string data to byte data using ASCII encoding.
               byte[] byteData =
 Encoding.ASCII.GetBytes(Command.CurrentValue);
               handler.Send(byteData);
               int bytesRec = handler.Receive(bytes);
               data += Encoding.ASCII.GetString(bytes, 0, bytesRec);
```

Installazione Plug-In

```
                ReturnString.CurrentValue = data;
                link.SetS("$return_string", ReturnString.CurrentValue);
                SendCommand.CurrentValue = -1;
                }
        //state_Locator
        if (Command.CurrentValue.Contains("state_Locator")) {
                // Convert the string data to byte data usingASCII encoding.
                byte[] byteData = Encoding.ASCII.GetBytes(Command.CurrentValue);
                handler.Send(byteData);  int bytesRec = handler.Receive(bytes);
                data +=  Encoding.ASCII.GetString(bytes, 0, bytesRec);
                ReturnString.CurrentValue = data;
                link.SetS("$return_string", ReturnString.CurrentValue);
                SendCommand.CurrentValue = -1;
        }
        //start_empty
        if (Command.CurrentValue.Contains("start_Empty")) {
                // Convert the string data to byte data using ASCII encoding.
                byte[] byteData = Encoding.ASCII.GetBytes(Command.CurrentValue);
                handler.Send(byteData);
                int bytesRec = handler.Receive(bytes);
                data += Encoding.ASCII.GetString(bytes, 0, bytesRec);
                ReturnString.CurrentValue = data;
link.SetS("$return_string",ReturnString.CurrentValue);
                SendCommand.CurrentValue = -1;
        }
        //stop_Locator
        if (Command.CurrentValue.Contains("stop_Locator")) {
                // Convert the string data to byte data using ASCII encoding.
                byte[] byteData = Encoding.ASCII.GetBytes(Command.CurrentValue);
                handler.Send(byteData);
                ReturnString.CurrentValue = "Ok";
                link.SetS("$return_string", ReturnString.CurrentValue);
        SendCommand.CurrentValue = -1;
        }
        //test_locator
        if (Command.CurrentValue.Contains("test_Locator")) {
                // Convert the string data to byte data using ASCII encoding.
                byte[] byteData =Encoding.ASCII.GetBytes(Command.CurrentValue);
                handler.Send(byteData);
                int bytesRec = handler.Receive(bytes);
                data += Encoding.ASCII.GetString(bytes, 0, bytesRec);
                ReturnString.CurrentValue = data;
link.SetS("$return_string",ReturnString.CurrentValue);
                SendCommand.CurrentValue = -1;
        }
```

Installazione Plug-In

```
        //start_Control
        if (Command.CurrentValue.Contains("start_Control")) {
                // Convert the string data to byte data using ASCII encoding.
                byte[] byteData = Encoding.ASCII.GetBytes(Command.CurrentValue);
                handler.Send(byteData);
                int bytesRec = handler.Receive(bytes);
                data += Encoding.ASCII.GetString(bytes, 0, bytesRec);
                ReturnString.CurrentValue = data;
link.SetS("$return_string",ReturnString.CurrentValue);
                SendCommand.CurrentValue = -1;
        }
        //start_Control
        if (Command.CurrentValue.Contains("start_Locator") ||
Command.CurrentValue.Contains("turn_Locator")) {
                Start_Locator:
                // Convert the string data to byte data using ASCII encoding.
                byte[] byteData = Encoding.ASCII.GetBytes(Command.CurrentValue);
                handler.Send(byteData);
                int bytesRec = handler.Receive(bytes);
                data = Encoding.ASCII.GetString(bytes, 0, bytesRec);
                if (data.Contains("Hopper")) {
                        var t = System.Threading.Tasks.Task.Run(() => ShowThreadInfo("Task"));
                        goto Start_Locator;
                }
                else {
                        if (data.Contains("#")) {
                                ReturnString.CurrentValue = data;
                                link.SetS("$return_string", ReturnString.CurrentValue);
                                SendCommand.CurrentValue = -1;
                        }
                        else {
                                string[] subs = data.Split(';');
                                // Set, read, and delete a location variable
                                Transform3D loc = new  Transform3D(Convert.ToDouble(subs[1]),
Convert.ToDouble(subs[2]), 180, 0, 180,  Convert.ToDouble(subs[3]));
                                link.SetL("pick_pos", loc);  Transform3D lVal = link.ListL("pick_pos");
                                SendCommand.CurrentValue= -1;
                                ReturnString.CurrentValue = "Ok Position";
                                link.SetS("$return_string", "Ok Position");
```

Installazione Plug-In

```
                                        if (lVal != loc)
                                            throw new
                                        InvalidOperationException();
                                    }
                                }
                            }
                        }
                    }
                    catch (Exception e) {
                        Trace.WriteLine(e.ToString());
                        //System.Threading.Thread.Sleep(200);
                    }
                    handler.Shutdown(SocketShutdown.Both);  handler.Close();
                }
                static void ShowThreadInfo(String s){
                    Trace.WriteLine(s);
                }
            }
            }
```

Lista Comandi FlexiVision

Per inviare il comando a FlexiVision è necessario modificare il valore della stringa "command".

N_Mission	Command	Action
1	“start_Locator"	Starts the parts localization process by recalling the FlexiBowl handling routine in case there are no parts that can be picked up. ***Return:*** “Pattern1;x;y;r”.
2	“stop_Locator"	Stops the process of locating the object with the aid of the FlexiBowl.
3	“turn_Locator"	If no parts are picked up, by this command the operator can make the Flexibowl rotate and the “start_Locator” routine start. ***Return:*** “Pattern1;x;y;r”.
4	“test_Locator"	Starts the process of locating the object without the aid of the FlexiBowl. ***Return:*** “Pattern1;x;y;r”.
5	“start_Control"	Starts the inspection cycle. ***Return:*** “Control1;x;y;r”.
6	“state_Locator"	Locator status diagnostics is shown: ***Return:*** “Locator is Running” “Locator is in Error” “Locator is not Running”.
7	“start_Empty"	Start the FlexiBowl® Quick-Emptying sequence. ***Return:*** “start_Empty ended”
8	“get_Recipe"	The name of the recipe currently loaded on FlexiVision is shown. ***Return:*** “recipe name”.
9	“set_Recipe=recipe name"	The recipe corresponding to the sent "recipe name" is loaded.