[메타캠프] FLEXING EDUKiT Manual

Contents

FLEXING EDUKiT?

공정 모델링과 시뮬레이션이 가능한 실습 환경과 실제 시스템 구축과 동일한 임베디드 디바이스, 액션 모듈, 대시보드 등을 통한 실습이 가능한 스마트팩토리 교육용 키트입니다.

1. SmartConnector

> SmartConnector란 EDUKiT의 데이터를 수집하기 위해 .net과 XGT Protocol으로 개발된 프로그램입니다.
> Sample Code는 말 그대로 Sample Code입니다. 자신이 필요한 동작 로직이 모두 Sample Code에 있다면 그대로 사용해도 됩니다. 다만, 원하는 동작 로직이 아니라면 제공된 코드를 수정하거나 새로 개발해야 합니다.

1.1 Sample Code

> Sample Code는 EDUKiT 데이터를 수집할 수 있는 코드가 들어가 있습니다.
> 코드를 실행시키기 위해서는 **.net core 3.1**이 자신의 PC에 설치되어 있어야합니다.

1.1.1 Sample Code 실행시키기

1. 제공된 폴더 경로에서 **FlexingEdukit\SmartConnector\bin\Debug\netcoreapp3.1** 폴더를 선택합니다.
2. **EdgeConfigFile.json** 파일은 Open합니다. (연결프로그램: 메모장, Notepad++ 등)

    2.1. 자신의 네트워크 환경에 맞게 파일을 수정하고 파일을 닫습니다.

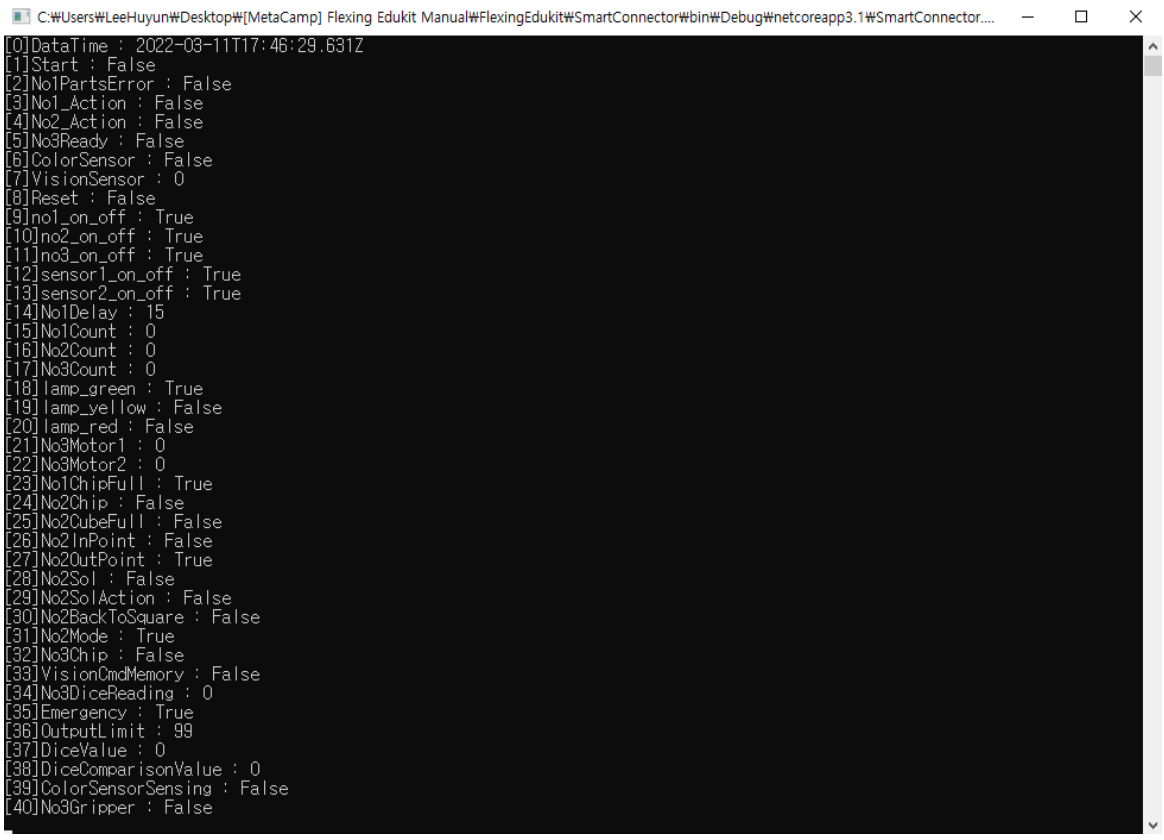
```
{
"EdukitId": "UVC-EDU-01",
"EdukitIP": "192.168.0.122",
"EdukitPort": "2004",
"MqttBrokerIP": "192.168.0.148",
"MqttBrokerPort": "1883",
"WebSocketServerUrl": "http://localhost:8282",
"DelayTime": "200",
"DebugType": "Debug"
}
```

Column	Comment
EdukitId	Mqtt Topic, Socket Id로 사용할 이름
EdukitIP	자신이 가지고 있는 EDUKiT의 IP
EdukitPort	자신이 가지고 있는 EDUKiT의 Port
MqttBrokerIP	MqttBroker의 IP
MqttBrokerPort	MqttBroker의 Port
WebSocketServerUrl	Socket Url
DelayTime	EDUKiT 데이터 수집 주기 (단위: ms)
DebugType	프로그램 Debug Type (Default: Debug)

3. 설정을 완료하였다면 **SmartConnector.exe**를 실행합니다.

   아래의 실행 결과가 나타면 성공적으로 데이터가 수집되고 있는 것을 확인할 수 있습니다.

   또한 MQTT Client로 해당 topic을 Subscribe하면 데이터가 들어오는 것을 확인할 수 있습니다.



1.1.2 Sample Code 실행이 안돼요!

1. **.net core 3.1이 깔려있는지 확인합니다.**

   깔려있지 않다면 제공된 파일의 **dotnet-sdk-3.1.416-win-x86.exe**를 클릭하여 다운로드 합니다.

2. **EdgeConfigFile.json의 문법이나 오타를 확인합니다.**

   json형식에 알맞게 작성했는지 확인합니다.

3. **MQTT Broker, EDUKiT의 주소가 일치한지 확인합니다.**

1.1.3 이 데이터는 어떤 데이터인가요?

- Sample Code를 실행했을 때 출력되는 데이터의 설명은 제공된 파일의 **FLEXING EDUKiT Sample Code 태그리스트**를 참조하면 됩니다.

2. SmartConnector Code

> SmartConnector의 구조와 데이터 수집 방법을 알아봅시다.
> SmartConnector를 수정하려면 Visiual Studio가 PC에 설치되어 있어야 합니다.

2.1 SmartConnector Code 수정하기

1. 제공된 파일의 FlexingEdukit 폴더에서 **FlexingEdukit.sln** 파일을 Open합니다.

2.2 프로젝트 구조

SmartConnector

┗ **EdgeConfigFile.json** [Edge 설정 파일]

┗ **program.cs** [프로젝트의 Main Code]

┗ **XGTAddressData.cs** [XGT를 사용하기 위한 Code]

┗ **XGTClass.cs** [XGT를 사용하기 위한 Code]

┗ **XGTData.cs** [XGT를 사용하기 위한 Code]

코드는 EdgeConfigFile.json과 Program.cs만 수정하면 됩니다.

2.3 Program.cs 함수 별 기능 설명

Function Name	Comment
Start	프로그램 시작 함수. SetConfig, Connect 함수를 호출함
SetConfig	EdgeConfigFile을 읽어오는 함수
Connect	Mqtt를 연결하는 함수. ConnectionStart 함수를 호출함
ConnectionStart	PLC를 연결하고 데이터를 수집하는 구문
ServerSocket_Unhandled	Socket Room를 Join하는 함수
SocketIoData	Socket으로 Data를 보내는 함수
MqttData	MQTT로 Data를 보내는 함수

2.3.1 ConnectionStart 함수 설명

> ConnectionStart 함수에서는 PLC 데이터 읽기/쓰기 로직을 실행시킵니다.

1. **PLC와 연결하기**

```
xGTClass.Connect(ip, port);
```

2. **PLC에 데이터 쓰기**

```
// 읽어올 데이터를 XGTAddressData로 생성
XGTAddressData pAddress2 = new XGTAddressData();

pAddress2.Address = "0";
pAddress2.Data = "0";

xGTClass.Write(XGT_DataType.Bit, pAddress2, XGT_MemoryType.SubRelay_M, 0);
```

3. **PLC에서 데이터 읽기**

```
// BitAddressData와 WordAddressData를 분리함
// 메모리주소의 알파벳이 M,P이면 Bit, D,C,K이면 Word임
Dictionary<XGTAddressData, string> BitAddressList = new Dictionary<XGTAddressData, string>();
Dictionary<XGTAddressData, string> WordAddressList = new Dictionary<XGTAddressData, string>();

// 읽어올 데이터를 XGTAddressData로 생성
XGTAddressData Start = new XGTAddressData();

// 생성한 XGTAddressData에 주소, 이름, TagId 입력
// Name과 TagId는 사용자가 원하는 Name과 TagId로 설정할 수 있음
Start.Address = "0"; // 메모리 주소중 숫자만 입력 (Ex) M00000이면 0만 입력
Start.Name = "Start";
Start.TagId = "1";

// 처음에 생성한 AddressList에 XGTAddressData와 메모리주소의 알파벳 부분을 추가함
// 알파벳에 따라 BitAddressList와 WordAddressData 중 알맞은 리스트에 넣어주어야 함
BitAddressList.Add(Start, "M");

// 데이터 수집 로직
while (true)
{
    try
    {
        // 위에서 추가한 태그 데이터를 담아놓을 리스트를 생성함
        List<EdukitNewdata> edukitData = new List<EdukitNewdata>();
        List<EdukitNewdata> edukitMqttData = new List<EdukitNewdata>();

        // BitAddressList를 읽어로는 로직 (M,P번지)
        foreach (var address in BitAddressList)
        {
            XGTData val = null;
            if (address.Value == "M")
            {
                val = xGTClass.Read(XGT_DataType.Bit, address.Key, XGT_MemoryType.SubRelay_M, 0);

                if (val.DataList != null)
                {
                    if (val.DataList[0].IntData == 0)
                    {
                        EdukitNewdata newdata = new EdukitNewdata
                        {
                            name = address.Key.Name,
                            tagId = address.Key.TagId,
                            value = false
                        };
                        edukitData.Add(newdata);
                    }
```

```
                else
                {
                    EdukitNewdata newdata = new EdukitNewdata
                    {
                        name = address.Key.Name,
                        tagId = address.Key.TagId,
                        value = true
                    };
                    edukitData.Add(newdata);
                }
            }
            Thread.Sleep(DelayTime / 100);
        }
        else if (address.Value == "P")
        {
            val = xGTClass.Read(XGT_DataType.Bit, address.Key, XGT_MemoryType.IO_P, 0);

            if (val.DataList != null)
            {
                if (val.DataList[0].IntData == 0)
                {
                    EdukitNewdata newdata = new EdukitNewdata
                    {
                        name = address.Key.Name,
                        tagId = address.Key.TagId,
                        value = false
                    };
                    edukitData.Add(newdata);
                }
                else
                {
                    EdukitNewdata newdata = new EdukitNewdata
                    {
                        name = address.Key.Name,
                        tagId = address.Key.TagId,
                        value = true
                    };
                    edukitData.Add(newdata);
                }
            }
            Thread.Sleep(DelayTime / 100);
        }
    }
    // BitAddressList를 읽어로는 로직 (D,C,K번지)
    foreach (var address in WordAddressList)
    {
        XGTData val = null;

        if (address.Value == "D")
        {
            val = xGTClass.Read(XGT_DataType.Word, address.Key, XGT_MemoryType.DataRegister_D, 0);

            if (val.DataList != null)
            {

                EdukitNewdata newdata = new EdukitNewdata();

                double data = (double)val.DataList[0].IntData;
                if (address.Key.Name == "No1Delay") data = (double)val.DataList[0].IntData / 10;

                newdata.name = address.Key.Name;
                newdata.tagId = address.Key.TagId;
                newdata.value = data.ToString();
                edukitData.Add(newdata);
            }
            Thread.Sleep(DelayTime / 100);
        }
        else if (address.Value == "C")
        {
            val = xGTClass.Read(XGT_DataType.Word, address.Key, XGT_MemoryType.Counter_C, 0);

            if (val.DataList != null)
            {
```

```
                EdukitNewdata newdata = new EdukitNewdata
                {
                    name = address.Key.Name,
                    tagId = address.Key.TagId,
                    value = val.DataList[0].IntData.ToString()
                };
                edukitData.Add(newdata);
            }
            Thread.Sleep(DelayTime / 100);
        }
        else if (address.Value == "K")
        {
            XGTData val1 = xGTClass.Read(XGT_DataType.Word, address.Key, XGT_MemoryType.KeepRelay_K, 0);
            int test = Int32.Parse(address.Key.Address);
            XGTAddressData q = new XGTAddressData
            {
                Address = (test + 1).ToString()
            };
            XGTData val2 = xGTClass.Read(XGT_DataType.Word, q, XGT_MemoryType.KeepRelay_K, 0);
            int val22 = val2.DataList[0].IntData;
            long dWordVal = (val22 * 65536) + val1.DataList[0].IntData;

            EdukitNewdata newdata = new EdukitNewdata
            {
                name = address.Key.Name,
                tagId = address.Key.TagId,
                value = dWordVal.ToString()
            };
            edukitData.Add(newdata);
            edukitMqttData.Add(newdata);

            Thread.Sleep(DelayTime / 100);
        }
    }
    // 0번째 태그에는 현재 시간을 추가함
    EdukitNewdata newdata2 = new EdukitNewdata
    {
        name = "DataTime",
        tagId = "0",
        value = DateTime.Now.ToString("yyyy'-'MM'-'dd'T'HH':'mm':'ss'.'fff'Z'")
    };
    edukitData.Add(newdata2);

    // Digital Twin에서 사용할 데이터로 Wrapper로 한번 감싸줌
    MqttData mqttData = new MqttData
    {
        Wrapper = edukitData
    };

    // Socket으로 데이터 송신하는 함수
    SocketIoData(edukitData);
    // Mqtt로 데이터 송신하는 함수
    MqttData(mqttData);

    // EdgeConfigFile.json에서 DebugType을 Debug로 설정했다면 Terminal에 데이터 Write
    if (edgeConfigResult.DebugType == "Debug")
    {
        Console.Clear();

        List<EdukitNewdata> SortedList = edukitData.OrderBy(x => Int32.Parse(x.tagId)).ToList();

        foreach (var data in SortedList)
        {
            Console.WriteLine($"[{data.tagId}]{data.name} : {data.value}");
        }
    }
}
catch (Exception ex)
{
    Console.WriteLine(ex.Message);

}
```

```
    // EdgeConfigFile.json에서 설정한 시간만큼 Delay
    Thread.Sleep(DelayTime);
}
```

2.4 수정한 코드 빌드하기

Ctrl + Shift + B를 눌러 프로젝트를 빌드합니다.

Visual Studio 하단 [출력] 부분에 성공이라고 출력되면 빌드에 성공한 것입니다.

만약 실패가 출력되면 실패한 이유를 보고 코드를 수정합니다.

빌드에 성공하면 **FlexingEdukit\SmartConnector\bin\Debug\netcoreapp3.1**에 있는 SmartConnector.exe를 실행합니다.

프로그램 동작을 보며 자신이 설계한 로직처럼 동작하는지 확인합니다.