표준 데이터 로거 센서 모듈(Sensor DLL) 제작 가이드

1. **들어가기 전에**

이 문서는 표준 데이터 로깅 프로그램에서 센서를 인식하고, 값을 측정 및 계산하는 DLL 형태의 센서 모듈(Sensor DLL)을 작성하는 방법을 설명합니다.

Sensor DLL 을 제작함으로, 다양한 종류의 센서를 표준 데이터 로거에 연결할 수 있습니다.

표준 데이터 로거에서 지원하는 센서 연결 방법은 다음과 같습니다.

* Analog Voltage, Current 센서
* RS232, RS485 로 통신하는 센서
* I2C 로 통신하는 센서
* Frequency, Gray Code, Pulse Count 등 Digital 센서

이 예제에서는 Analog Voltage 타입의 센서를 하나 연결해 볼 것입니다.

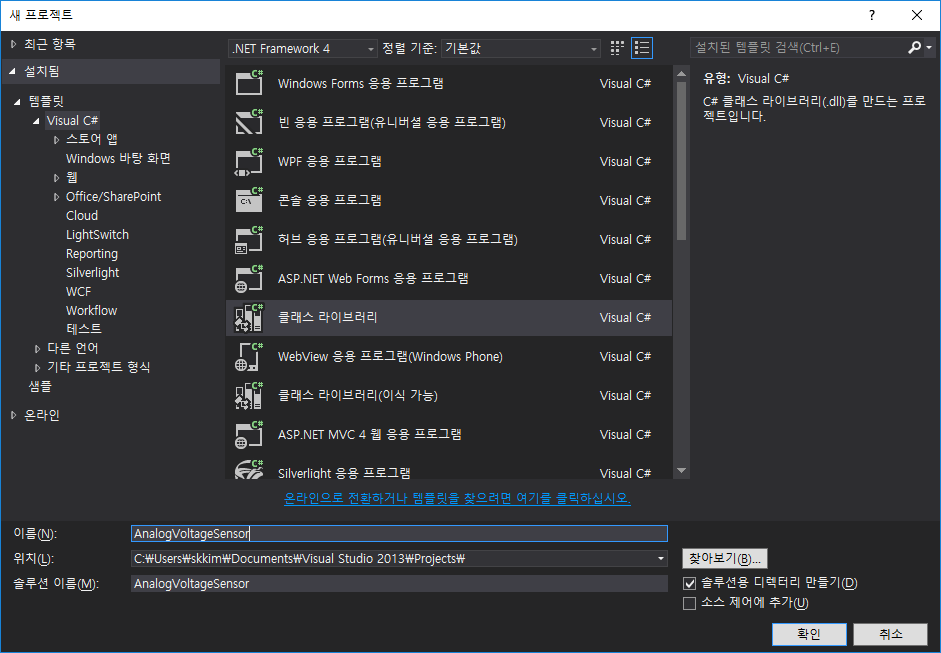
표준 데이터 로깅 프로그램은 C# 언어로 제작되었으며, .NetFramework 4.0 환경으로 개발되었습니다.

1. **개발 환경 준비**

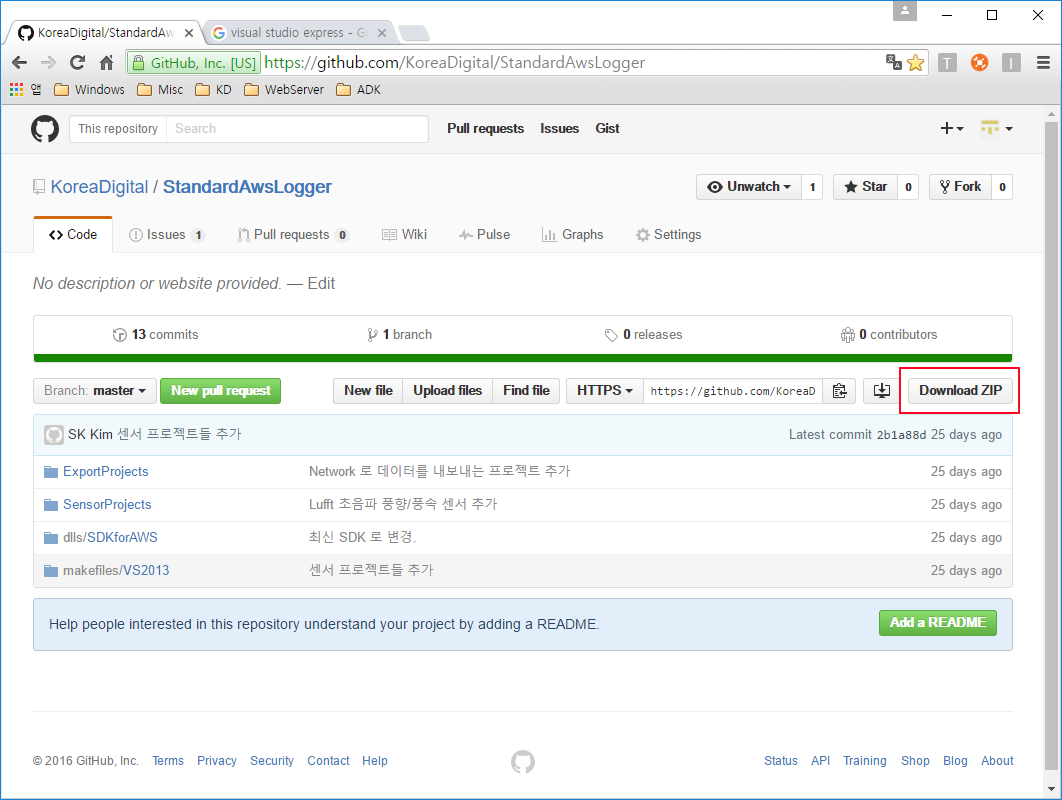
OS : Windows Vista 이상 버전이 필요합니다.  
IDE : Microsoft Visual Studio 2010 이상 버전이 필요합니다.(Express Edition, Community Edtion 가능)

1. **프로젝트 생성**

Visual Studio 를 실행하고, C# 새 프로젝트를 생성합니다. 대상 Framework 는 4.0, 프로젝트 형식은 클래스 라이브러리로 지정하고 프로젝트 이름을 적습니다.

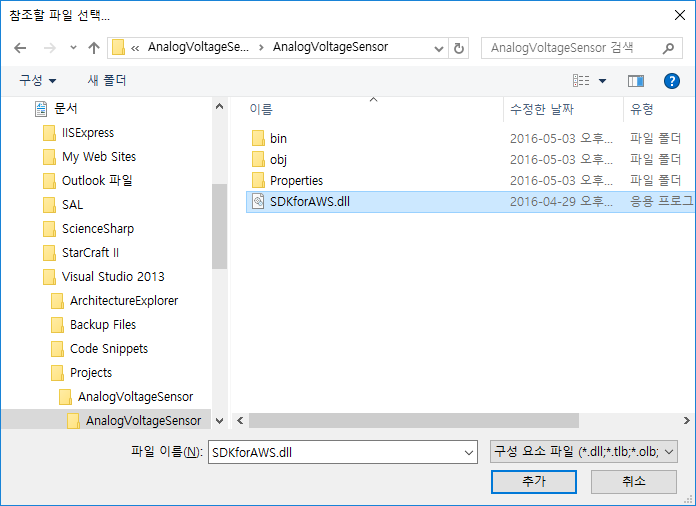
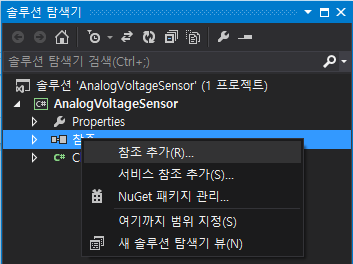


1. **소스코드 다운로드**

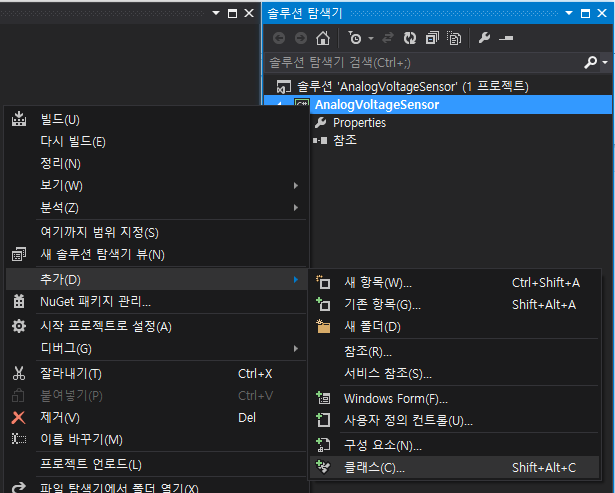
표준 데이터 로거의 GutHub 오픈 소스 프로젝트 사이트(<https://github.com/KoreaDigital/StandardAwsLogger>)에 접속하여, 소스코드를 다운로드 합니다.  
  
다운로드한 소스파일의 압축을 해제하고,  
StandardAwsLogger-master\dlls\SDKforAWS\Debug\ 아래 있는 SDKforAWS.dll 과 SDKforAWS.pdb 파일을 새로 생성한 프로젝트 폴더로 복사합니다.

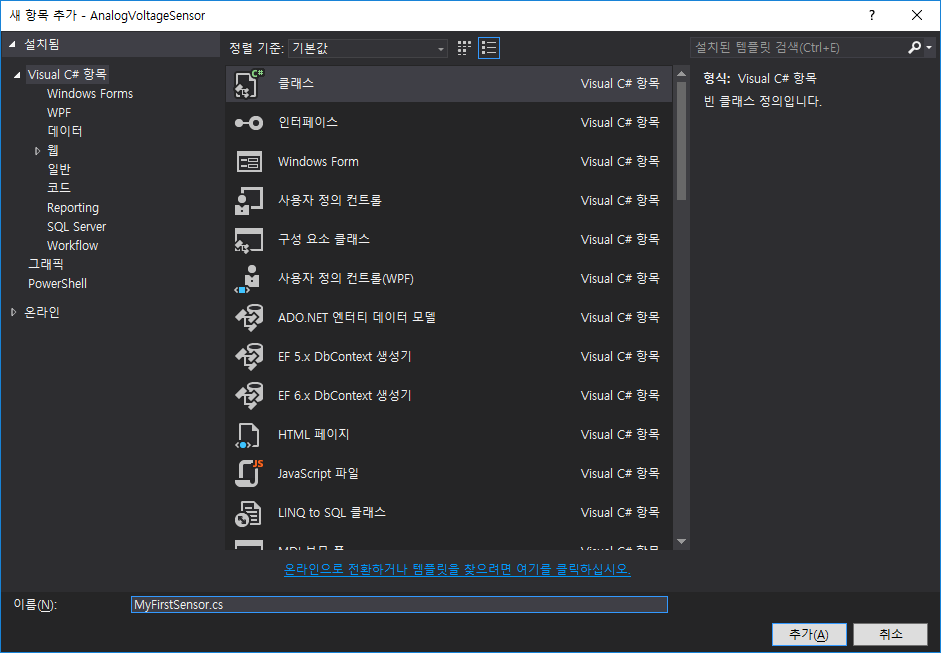
1. **프로젝트에 SDK DLL 참조 추가**

Visual Studio 에서 솔루션 탐색기 창을 열고, 생성한 프로젝트의 참조에서 참조 추가를 선택합니다.  
  
Step 3 에서 복사한 DLL 파일을 선택합니다.



1. **클래스 파일 추가하기**

프로젝트를 생성할 때, 자동 생성된 Class1.cs 파일은 삭제하고, MyFirstSensor 클래스를 추가합니다.  




1. **코드 작성하기**

SDK 를 사용하기 위한 using 선언문을 추가합니다.

using SDKforAWS;

Sensor 모듈을 만들기 위해, 생성한 MyFirstSensor 클래스가 SDKforAWS의 WSensor 를 상속하도록 합니다. WSensor 는 내보내기 모듈의 동작을 정의하는 추상 클래스 입니다.

public class MyFirstSensor : WSensor

WSensor 클래스는 다음과 같이 정의되어 있습니다.

namespace SDKforAWS

{

public abstract class WSensor

{

public string sensor\_name;

public string description;

public SensorInputType input\_type;

public List<WSensor.Unit> units;

public double maxsamplingHz;

public double minsamplingHz;

protected WInterface Interface;

... 중략 ...

public WSensor(int modulnum, int channelnum);

public abstract bool OnInitialize();

public abstract void OnDestroy();

public abstract bool OnStart();

public abstract void OnStop();

public abstract void OnMeasure();

public abstract void Setup();

... 중략 ...

public class Unit

{

public KMASensorType sensor\_type;

public string unit\_name;

public string unit\_text;

public string unit\_format;

public double max;

public double min;

public double value;

public string format(bool isaddunit);

}

}

}

주요 함수들을 보면 다음과 같습니다.

|  |  |
| --- | --- |
| 함수 원형 | bool OnInitialize() |
| 설명 | 센서 모듈이 초기화 될 때, 호출되는 콜백 함수입니다. 센서 모듈당 1회만 호출되는 것을 보장합니다. 모듈 내에서 사용할 리소스들은 이 함수 내에서 초기화 해야 합니다. |
| 함수 원형 | void OnDestroy() |
| 설명 | 센서 모듈이 해제 될 때, 호출되는 콜백 함수입니다. 센서 모듈당 1회만 호출되는 것을 보장합니다. 모듈 내에서 사용했던 리소스들은 이 함수 내에서 해제해야 합니다. |
| 함수 원형 | bool OnStart() |
| 설명 | 센서에서 값을 측정하기 시작할 때, 호출되는 콜백 함수입니다. 사용자가 측정을 시작/정지를 반복하면 여러 번 호출 될 수 있습니다. 측정 중에만 사용되는 리소스들을 초기화나, 통신 연결 등은 이 함수 내에서 호출해줘야 합니다. |
| 함수 원형 | void OnStop() |
| 설명 | 센서에서 값 측정을 멈출 때, 호출되는 콜백 함수입니다. 사용자가 측정을 시작/정지를 반복하면 여러 번 호출 될 수 있습니다. 측정 중에만 사용되는 리소스들의 해제나, 통신 종료 등은 이 함수 내에서 호출해줘야 합니다. |
| 함수 원형 | void OnMeasure() |
| 설명 | 사용자가 지정한 Interval 간격으로, 센서에서 값을 측정할 때마다 호출되는 콜백 함수입니다. 이 함수 내에서, 센서로부터 값을 읽어서, 로깅 프로그램에 전달해줘야 합니다. 시간이 오래 걸리는 작업은 이 함수내에서 호출해서는 안되고, 별도의 Thread 를 만들어서 수행해야 합니다. |
| 함수 원형 | void Setup(); |
| 설명 | 사용자가 모듈을 추가할 때, 모듈별 설정값들을 지정할 때 호출되는 콜백 함수입니다. UI Thread 에서 호출되므로, WinForm 을 사용해도 됩니다. 주로 Form 을 띄우고, 사용자로부터 설정값 입력을 받습니다. |

Sensor 모듈은 정수형 구분자를 2개 받는 생성자를 제공해야 하며, WSensor 의 동일한 생성자를 호출해줘야 합니다.

public MyFirstSensor(int n1, int n2)

: base(n1, n2)

{

}

또한 생성자 내에서는, 센서의 연결 방법 및 센서 이름이나 설명, 측정값의 포맷 등을 지정할 수 있습니다. 위의 생성자의 안쪽에 다음 코드를 추가합니다.

public MyFirstSensor(int n1, int n2)

: base(n1, n2)

{

this.input\_type = SensorInputType.ANALOG\_SINGLE\_ENDED\_10V;

this.minsamplingHz = 1;

this.sensor\_name = "Voltage(0~5V)";

this.description = "Analog Voltage";

WSensor.Unit unit = new WSensor.Unit();

unit.unit\_name = "Voltage";

unit.unit\_text = "V";

unit.unit\_format = "{0:#0.000000}";

unit.min = 0;

unit.max = 5;

this.units.Add(unit);

}

이 예제에서 만들고자 하는 센서 모듈은 모듈 초기화 / 해제 시에 수행해야 하는 동작이 없습니다. 빈 함수로 남겨둡니다.

public override bool OnInitialize()

{

return true;

}

public override void OnDestroy()

{

}

측정 시작과 종료 시에 초기화 및 해제해야 하는 것도 없습니다. 빈 함수로 남겨둡니다.

public override bool OnStart()

{

return true;

}

public override void OnStop()

{

}

사용자 설정 또한 필요치 않습니다. 빈 함수로 남겨둡니다.

public override void Setup()

{

}

측정시마다, Voltage 값을 읽어서 로깅 프로그램에 전달해야 합니다.

public override void OnMeasure()

{

units[0].value = Interface.analog.getVoltage();

}

Interface.analog 는 Analog 센서를 연결하는 보드와 통신하기 위해 SDK 에서 제공하는 통신 인터페이스 입니다. Interface 객체에는 analog 외에도 digital, comm 등의 여러가지 통신 인터페이스 들이 포함되어 있습니다. 이 예제에서는 Analog 보드에 연결된 센서의 Voltage 값만 읽으면 되므로, 위의 코드와 같이 간단하게 표현됩니다.

완성된 전체 소스입니다.

using SDKforAWS;

namespace AnalogVoltageSensor

{

public class MyFirstSensor : WSensor

{

public MyFirstSensor(int n1, int n2)

: base(n1, n2)

{

this.input\_type = SensorInputType.ANALOG\_SINGLE\_ENDED\_10V;

this.minsamplingHz = 1;

this.sensor\_name = "Voltage(0~5V)";

this.description = "Analog Voltage";

WSensor.Unit unit = new WSensor.Unit();

unit.unit\_name = "Voltage";

unit.unit\_text = "V";

unit.unit\_format = "{0:#0.000000}";

unit.min = 0;

unit.max = 5;

this.units.Add(unit);

}

public override bool OnInitialize()

{

return true;

}

public override bool OnStart()

{

// \_\_noop;

return true;

}

public override void OnMeasure()

{

units[0].value = Interface.analog.getVoltage();

}

public override void OnStop()

{

}

public override void OnDestroy()

{

}

public override void Setup()

{

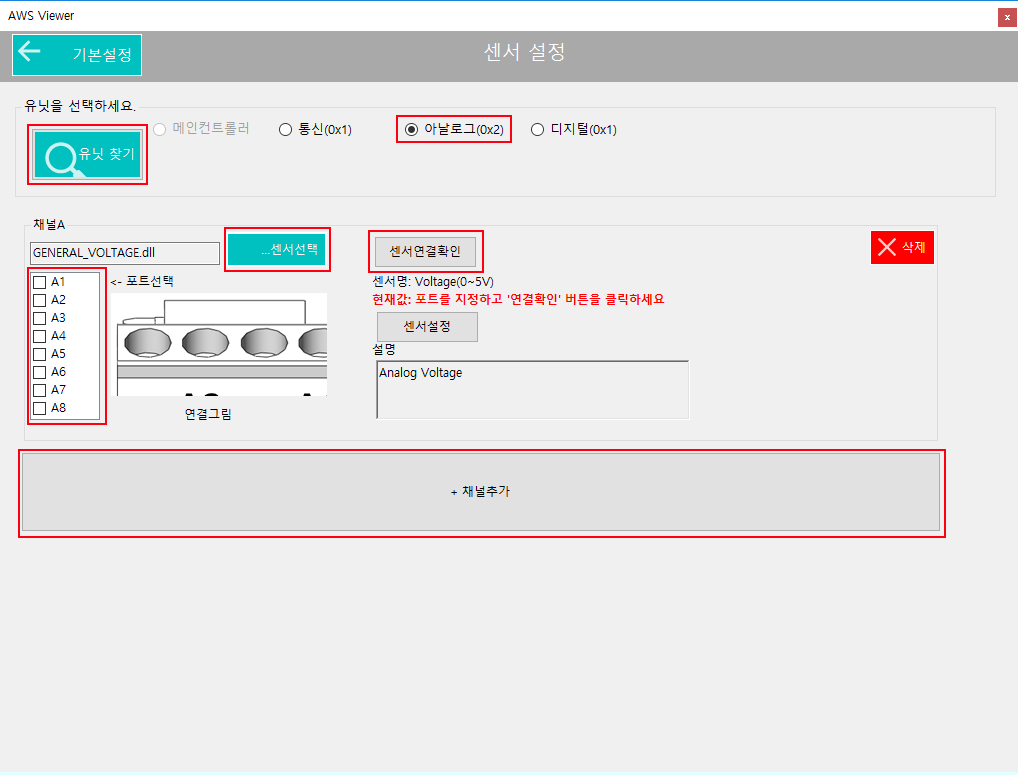
}

}

}

1. **실행**

프로젝트를 빌드하고, 프로젝트 폴더\bin\release\AnalogVoltageSensor.dll 파일을 로깅 프로그램과 동일한 폴더에 넣고, 로깅 프로그램을 실행합니다.  
로깅 프로그램의 센서 설정에서, 유닛 찾기 버튼을 눌러, 표준 데이터 로거에 연결되어 있는 유닛들은 찾습니다. 아날로그 유닛을 선택하고, 채널 추가 버튼을 눌러서, 새로운 채널을 하나 추가합니다.  
센서 선택 버튼을 눌러서, 제작한 Sensor DLL 을 선택하고, Analog Voltage 센서를 연결한 물리적 포트를 지정해줍니다.  
센서 연결 확인 버튼을 눌러서, 센서 값이 정상적으로 올라오고 있는지 확인합니다.



모니터링 화면으로 돌아가서, 데이터를 수집하여 센서값이 정상적으로 로깅 되는지 확인합니다.