Przemysłowe Sieci Komputerowe

Biblioteki dla języków Python/C# do komunikacji ze sterownikami S7-1200

Paweł Strączyński

p.straczynski@tu.kielce.pl

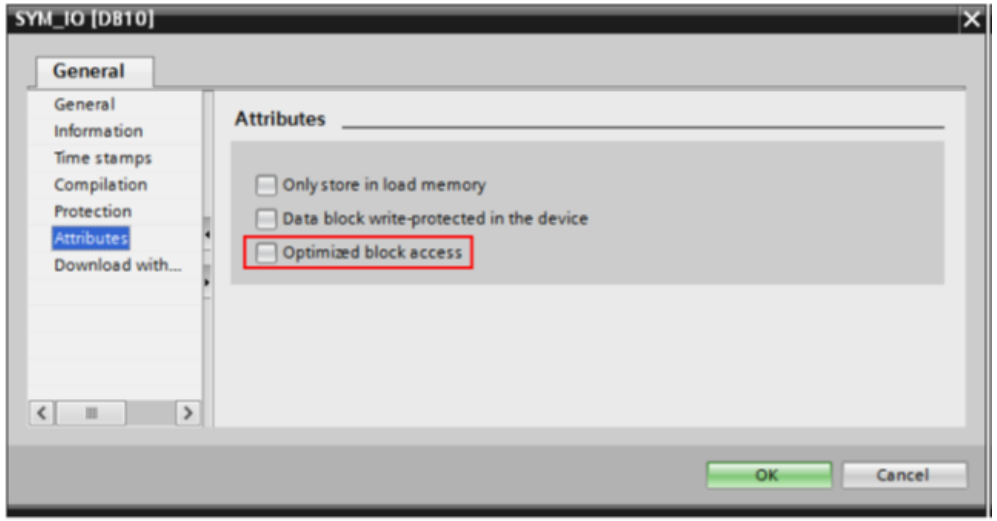
Katedra Urządzeń Elektrycznych i Automatyki

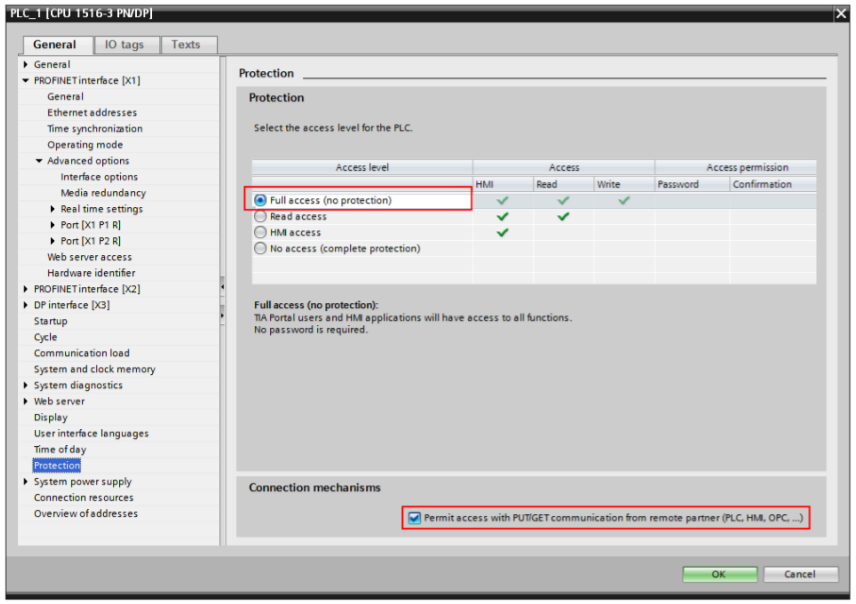
Politechnika Świętokrzyska

Dostęp do pamięci sterownika PLC Siemens S7

Dostęp do pamięci sterowników PLC Siemens S7 możliwy jest z wykorzystaniem specjalizowanych bibliotek dostępnych dla wielu języków programowania.

Przykład stanowić może pakiet bibliotek S7.Net dla języka C (https://github.com/S7NetPlus/s7netplus/wiki) lub rodzina bibliotek Snap7 która posiada swą implementacje m.in. w takich językach jak C/C++, Pascal, Python, Node.js, .NET/Mono(CVB.NET) (http://snap7.sourceforge.net). Istnieją również natywne wersje tej biblioteki dla języków C# (sharp7) oraz Java (Mokka7).





Biblioteka dla języka C# - S7.Net

Bibliotekę można pobrać z oficjalnego repozytorium na GitHub'ie:

https://github.com/killnine/s7netplus

lub bezpośrednio za pomocą NuGet'a:

https://www.nuget.org/packages/S7netplus/

S7.Net jest biblioteką dla sterowników PLC Siemens S7 komunikujących się za pomocą Ethernet'u.

S7.Net jest kompatybilne z S7-300, S7-400, S7-1200, S7-1500.

Biblioteka dla języka C# - S7.Net

Tworzenie instancji PLC

W celu utworzenia instancji PLC należy utworzyć obiekt klasy Plc:

```
public Plc(CpuType cpu, string ip, Int16 rack, Int16 slot)
```

Gdzie parametry to odpowiednio:

- `cpu` – typ podłączonego sterownika PLC
- `ip` – adres IP sterownika PLC
- `rack, slot` – konfiguracja sprzętowa w Step7/TIA Portal

```
public enum CpuType {
 S7200 = 0,
 S7300 = 10,
 S7400 = 20,
 S71200 = 30,
 S71500 = 40,
}
```

```
Plc plc = new Plc(CpuType.S7300, "127.0.0.1", 0, 2);
```

Biblioteka dla języka C# - S7.Net

Podłączenie i odłączenie od sterownika

Otwarcia połączenia dokonujemy z wykorzystaniem metody **Open()**

```
public ErrorCode Open();
```

Analogicznie zamknięcia połączenia dokonujemy z wykorzystaniem metody **Close()**

```
public void Close();
```

Ponadto aby sprawdzić status połaczenia możemy użyć atrybutów:

```
public bool IsAvailable;
public bool IsConnected;
```

```
public enum ErrorCode
{
 NoError = 0,
 WrongCPU_Type = 1,
 ConnectionError = 2,
 IPAddressNotAvailable,
 WrongVarFormat = 10,
 WrongNumberReceivedBytes = 11,
 SendData = 20,
 ReadData = 30,
 WriteData = 50
}
```

Biblioteka dla języka C# - S7.Net

Odczyt/zapis bajtu

Metody do odczytu i zapisu bajtu (maksymalnie 200 bajtów – aktualny limit protokołu):

```
public byte[] ReadBytes(DataType dataType, int db, int startByteAdr, int count);
public ErrorCode WriteBytes(DataType dataType, int db, int startByteAdr, byte[] value);
```

Znacznie poszczególnych argumentów:

- `dataType` – określa rodzaj pamięci do zapisu/odczytu,
- `db` – adres dla określonego typu danych,
- `startByteAdr` – adres pierwszego bajtu do odczytu/zapisu
- `Count` – ilość bajtów do odczytu/zapisu,
- `Value[]` – wartości do zapisu

```
public enum DataType
{
 Input = 129,
 Output = 130,
 Memory = 131,
 DataBlock = 132,
 Timer = 29,
 Counter = 28
}
```

Odczyt pierwszych 200 bajtów z DB1:

```
var bytes = plc.ReadBytes(DataType.DataBlock, 1,0,200);
```

Biblioteka dla języka C# - S7.Net

Odczyt/zapis różnych typów danych

Metody do odczytu:

```
public object Read(DataType dataType, int db, int startByteAdr, VarType varType, int varCount);
public ErrorCode Write(DataType dataType, int db, int startByteAdr, object value);
```

Znacznie poszczególnych argumentów:

- `dataType` – określa rodzaj pamięci do zapisu/odczytu,
- `db` – adres dla określonego typu danych,
- `startByteAdr` – adres pierwszego bajtu do odczytu/zapisu
- `Count` – ilość bajtów do odczytu/zapisu,
- `Value[]` – wartości do zapisu
- `VarType` – typ danej

```
public enum VarType
{
Bit,
Byte,
Word,
DWord,
Int,
DInt,
Real,
String,
Timer,
Counter
}
```

```
public enum DataType
{
 Input = 129,
 Output = 130,
 Memory = 131,
 DataBlock = 132,
 Timer = 29,
 Counter = 28
}
```

Odczyt 20 rejestrów typu DWord

```
var dwords = plc.Read(DataType.DataBlock, 1,0, VarType.DWord, 20);
```

Biblioteka dla języka C# - S7.Net

Odczyt/zapis pojedynczej zmiennej

Metody do odczytu pojedynczej zmiennej:

```
public object Read(string variable);
public ErrorCode Write(string variable, object value);
```

- **`variable`** – odczytywana/zapisywana zmienna zapisana w formie string'a np. „DB1.DBW20", „T45" „„C21"

Przykład odczytu zmiennej DB1.DBW0

```
ushort result = (ushort)plc.Read("DB1.DBW0");
```

Biblioteka dla języka C# - S7.Net

Odczyt/zapis struktur, klas

```
public object ReadStruct(Type structType, int db, int startByteAdr = 0)
public ErrorCode WriteStruct(object structValue, int db, int startByteAdr = 0)
```

- structType: Type of the struct to be read, for example: typeOf(MyStruct))
- db: index of the DB to read
- startByteAdr: specified the first address of the byte to read (the default is zero).

Example:

You define a DataBlock in the plc like:

Adresse	Name	Typ	Anfangswert
0.0		STRUCT	
+0.0	varBool0	BOOL	FALSE
+0.1	varBool1	BOOL	FALSE
+0.2	varBool2	BOOL	FALSE
+0.3	varBool3	BOOL	FALSE
+0.4	varBool4	BOOL	FALSE
+0.5	varBool5	BOOL	FALSE
+0.6	varBool6	BOOL	FALSE
+1.0	varByte0	BYTE	B#16#0
+2.0	varByte1	BYTE	B#16#0
+4.0	varWord	WORD	W#16#0
+6.0	varReal	REAL	1.230000e+000
+10.0	varBool7	BOOL	TRUE
+12.0	varReal1	REAL	8.506780e+002
+16.0	varbyte2	BYTE	B#16#55
+18.0	varDWord	DWORD	DW#16#1234567
=22.0		END_STRUCT	

```
public void ReadClass(object sourceClass, int db, int startByteAdr = 0)
public ErrorCode WriteClass(object classValue, int db, int startByteAdr = 0)
```

- sourceClass: instance of the class that you want to assign the values
- db: index of the DB to read
- startByteAdr: specified the first address of the byte to read (the default is zero).

Example:

You define a DataBlock in the plc like:

Adresse	Name	Typ	Anfangswert
0.0		STRUCT	
+0.0	varBool0	BOOL	FALSE
+0.1	varBool1	BOOL	FALSE
+0.2	varBool2	BOOL	FALSE
+0.3	varBool3	BOOL	FALSE
+0.4	varBool4	BOOL	FALSE
+0.5	varBool5	BOOL	FALSE
+0.6	varBool6	BOOL	FALSE
+1.0	varByte0	BYTE	B#16#0
+2.0	varByte1	BYTE	B#16#0
+4.0	varWord	WORD	W#16#0
+6.0	varReal	REAL	1.230000e+000
+10.0	varBool7	BOOL	TRUE
+12.0	varReal1	REAL	8.506780e+002
+16.0	varbyte2	BYTE	B#16#55
+18.0	varDWord	DWORD	DW#16#1234567
=22.0		END_STRUCT	

Biblioteka Snap7

Bibliotekę można pobrać z SourceForge'a:

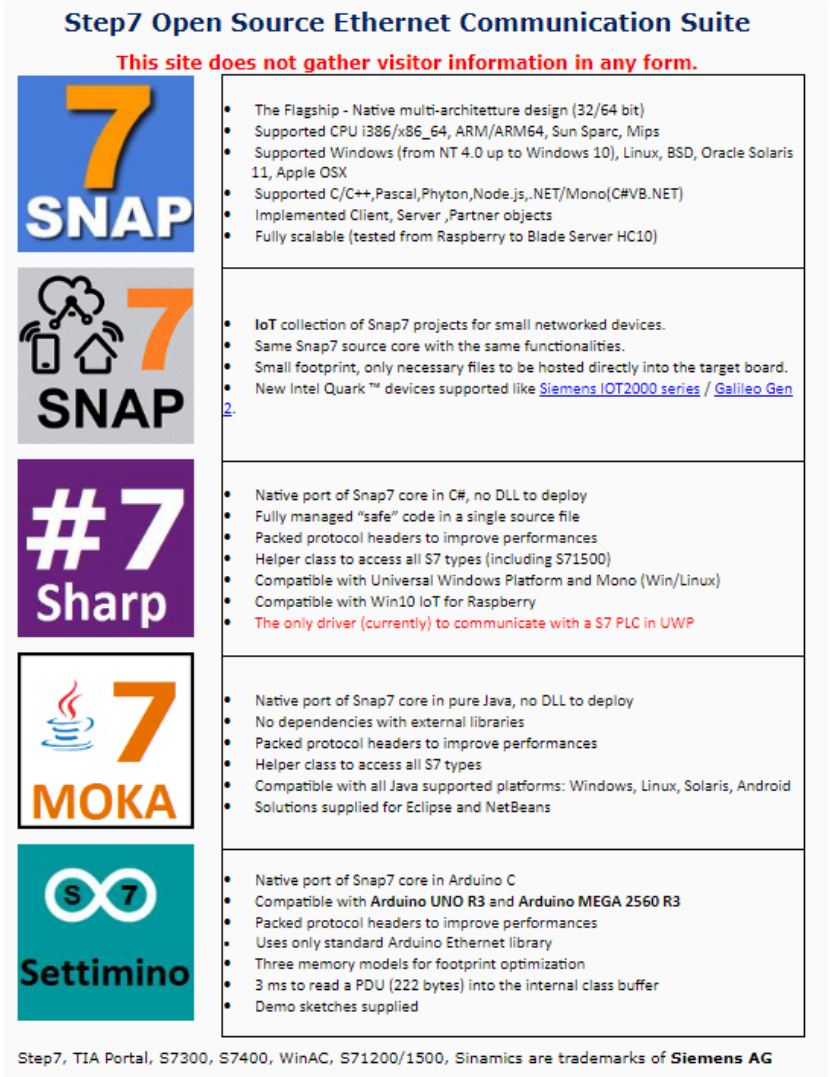
https://snap7.sourceforge.net/

lub w Pythonie z PIP'a:

pip install python-snap7

Snap7 jest biblioteką dla sterowników PLC Siemens S7 dla języków C/C++,Pascal, Phyton, Node.js, .NET/Mono(C#VB.NET

Biblioteka Snap7



Snap7 client - przykłady

```
>>> import snap7
>>> client = snap7.client.Client()
>>> client.connect("127.0.0.1", 0, 0, 1012)
>>> client.get_connected()
True
>>> data = client.db_read(1, 0, 4)
>>> data bytearray(b"\x00\x00\x00\x00")
>>> data[3] = 0b00000001
>>> data
bytearray(b'\x00\x00\x00\x01’)
>>> client.db_write(1, 0, data)
```

```
>>> import snap7
>>> client = snap7.client.Client()
>>> client.connect("192.168.0.1", 0, 0)
>>> buffer = client.db_get(1) # reads the db number 1.
>>> buffer
bytearray(b"\x00\x00\x00\x00\x00\x00\x00\x00...\x00\x00")
```

Snap7 client - przykłady

```
import snap7
import snap7.client as s7
from snap7.util import *
from snap7.snap7types import *
plc = s7.Client ()
Plc.connect("192.168.0.3",0, 1) # Odczyt danych z sterownika PLC - DB4007 - DBX0 .0
result = plc.read_area( S7AreaDB, 4007, 0, S7WLReal )
value = get_real( result ,0)
print( value ) #Zapis danych do sterownika PLC - DB4007 - DBX0 .0
result = ( wordlen_to_ctypes [ S7WLByte ]*10)()
set_real( result, 0, 15.0)
plc.write_area( S7AreaDB, 4007, 6 , result )
```

Snap7 client - przykłady

S7WLBit	0x01	Bit (inside a word)
S7WLByte	0x02	Byte (8 bit)
S7WLWord	0x04	Word (16 bit)
S7WLDWord	0x06	Double Word (32 bit)
S7WLReal	0x08	Real (32 bit float)
S7WLCounter	0x1C	Counter (16 bit)
S7WLTimer	0x1D	Timer (16 bit)

S7Consts.S7AreaPE	0x81	Process Inputs.
S7Consts.S7AreaPA	0x82	Process Outputs.
S7Consts.S7AreaMK	0x83	Merkers.
S7Consts.S7AreaDB	0x84	DB
S7Consts.S7AreaCT	0x1C	Counters.
S7Consts.S7AreaTM	0x1D	Timers